QGIS Desktop 3.28 User Guide

QGIS Project

2024 年 03 月 03 日

目 次

第1章　序文

このドキュメントは、地理情報システム（GIS）ソフトウェア「QGIS」のためのユーザーガイドです。QGIS は GNU General Public License に従います。より詳しい情報は、QGIS のホームページ https://www.qgis.org にあります。

このドキュメントの内容は、執筆者と編集者の知識の及ぶ限りの最善を尽くして執筆され、検証されています。しかし、そうではあってもやはり誤りがある可能性があります。

従って、執筆者、編集者および出版者は、このドキュメントの間違いとそれが引き起こしうる結果について、いかなる責任も負わないものとします。間違いと思われるものを見つけたときには、ご報告ください。

このドキュメントは reStructuredText を使って組版されています。これは、 github では reST ソースコードの形で、また、 https://www.qgis.org/en/docs/ からは HTML および PDF としてオンラインで入手可能です。このドキュメントの翻訳バージョンも同様に、QGIS プロジェクトのドキュメントエリアから閲覧とダウンロードが可能です。

このドキュメントに貢献したい場合や、翻訳についてのより詳しい情報は、https://qgis.org/en/site/getinvolved/index.html を参照してください。

ドキュメント内のリンクについて

このドキュメントには内部リンクと外部リンクが含まれています。内部リンクをクリックした場合はこのドキュメント内部を移動しますが、外部リンクをクリックした場合にはそのインターネットアドレスを開きます。

ドキュメントの執筆者と編集者

このドキュメントの執筆、レビュー、翻訳に貢献してきた人々のリストは、 **協力者** で見ることができます。

Internet: https://www.qgis.org

このドキュメントのライセンス

1.1 QGIS 3.28 の新機能

QGIS のこのリリースでは、 QGIS 3.28 と比較して、いくつものバグ修正と多数の新機能の追加および機能の拡張が行われています。新機能の完全なリストについては、ビジュアルチェンジログ https://qgis.org/en/site/forusers/visualchangelogs.html を参照してください。

第2章　はじめに

地理情報システム (GIS) のすばらしい世界へようこそ!

QGIS はオープンソースの地理情報システムです。プロジェクトは 2002 年 5 月に誕生し、同年 6 月に SourceForge 上でプロジェクトとして設立されました。私たちは、（従来は高価なプロプライエタリソフトウェアであった） GIS ソフトウェアを、パーソナルコンピューターにアクセスできる人なら誰でも利用できるようにするために努力してきました。

QGIS は現在、ほとんどの Unix プラットフォーム、Windows、macOS 上で動作します。 QGIS は Qt ツールキット (https://www.qt.io) と C++ を使用して開発されています。このため、 QGIS の動作は軽快で、使いやすいグラフィカルユーザーインタフェース (GUI) を備えています。また、 QGIS を現場に持ち出すことを可能にする、独自に作成されたアプリケーションもあります。これらのアプリケーションは Android や iOS 上で動作します。

QGIS はユーザーフレンドリーな GIS になることを目指し、広く使われる機能を提供しています。このプロジェクトの最初の目的は、 GIS データのビューアを提供することでした。 QGIS は進化を続け、今では日々の GIS データの閲覧、データの取得、高度な GIS 分析、洗練された地図や地図帳、レポートによるプレゼンテーションに使われるようになっています。 QGIS は豊富なラスタフォーマットとベクタフォーマットをサポートするとともに、プラグインアーキテクチャを利用することにより、新しいフォーマットへのサポートも簡単に追加することができるようになっています。

QGIS は GNU General Public License (GPL) の下でリリースされています。QGIS がこのライセンスの下で開発されていることは、ソースコードを調べてこれに変更を加えることができることを意味し、ユーザーに対しては、いつでも無料で変更が自由に可能な GIS プログラムにアクセスできることを保証します。QGIS を入手する際にライセンスの完全なコピーも同時に受け取っているはずですが、このドキュメントの **付録** *A: GNU* **一般公衆ライセンス（***General Public License***）** のセクションから入手することもできます。

Tip: 最新版のドキュメントについて

このドキュメントの最新版は、 QGIS ウェブサイトのドキュメントエリア https://www.qgis.org/ja/docs/ でいつでも確認することができます。

第3章　記述ルール

このセクションでは、このマニュアルを通して使われている、一貫した記述ルールについて説明します。

3.1 GUI 記述ルール

グラフィカルユーザインタフェース (GUI) の記述スタイルは、GUI の見た目に似せるように意図されています。このスタイルは一般的にツールチップが表示されていない状態を反映しています。このためユーザは GUI の外観をざっと眺めるだけで、マニュアルの指示と同じものを見つけることができます。

- メニューのオプション：**レイヤ　ラスタレイヤの追加** や **設定　ツールバー　デジタル化**
- ツール：  ラスタレイヤの追加
- ボタン：**デフォルトとして保存**
- ダイアログボックスの見出し：**レイヤプロパティ**
- タブ：**一般情報**
- チェックボックス  **描画**
- ラジオボタン：  *Postgis SRID*  *EPSG ID*
- 数値を選択： 
- 文字を選択： 
- ファイルの一覧から選択：...
- 色の選択： 
- スライダー： 
- テキストの入力： 

影がついているものは、クリック可能な GUI コンポーネントであることを表しています。

3.2 テキストやキーボードの記述ルール

このマニュアルでは、クラスやメソッドなど異なるエンティティを識別するために、テキストやキーボードショートカット、コーディングに関連するスタイルも使われています。これらのスタイルは QGIS での実際のテキストやコードの外観には一切対応していません。

- ハイパーリンク: https://qgis.org
- キーの組み合わせ：`Ctrl+B` を押す ： これは Ctrl キーを押したまま、B キーを押すことを意味します。
- ファイル名： `lakes.shp`
- クラス名： **NewLayer**
- メソッド： *classFactory*
- サーバー： *myhost.de*
- ユーザー入力テキスト： `qgis --help`

プログラムのコードは固定幅フォントで表示されます。

```
PROJCS["NAD_1927_Albers",
  GEOGCS["GCS_North_American_1927",
```

3.3 プラットフォーム固有の指示

GUI でのひとつづきの操作で、文字も少ない場合は、インラインで（行内で）フォーマットされます。たとえば： **ファイル** *QGIS* **終了して** *QGIS* **を閉じる** をクリック– – これは「Linux、Unix および Windows プラットフォーム上では最初に [ファイル] メニューをクリックしてそれから [終了] する」こと、一方「macOS プラットフォーム上では最初に [QGIS] メニューをクリックしてそれから [終了] する」ことを示しています。

文字が多くなる場合はリストとしてフォーマットされることもあります。

- Linux・Unix ではこうします
- Windows ではこうします
- macOS ではこうします

または、段落としてフォーマットされることもあります。

これは Linux・Unix・macOS プラットフォーム向けの解説です。文章中の解説手順に基づいて作業してください。

これは Windows プラットフォーム向けの解説です。文章中の解説手順に基づいて作業してください。

ユーザーガイド中のスクリーンショットはいろいろなプラットフォームで作成されています。

第4章　QGISの主な機能

QGIS はコア機能とプラグインにより豊富な GIS 機能を提供しています。ロケータバーが機能やデータセットやその他の検索を容易にしています。

主要機能とプラグインが構成する 6 つの総合的なカテゴリについて、以下に簡単にまとめます。また付属の Python コンソールについても簡単に案内します。

4.1 データの閲覧

様々なフォーマットと投影法による平面もしくは 3 D のベクタデータおよびラスタデータを、内部フォーマットや共通フォーマットへ変換することなく、組み合わせて閲覧することができます。サポートされているフォーマットは以下のものです。

- PostGIS や SpatiaLite、MS SQL Spatial を使用した空間属性をもつテーブルやビュー、Oracle Spatial、インストールされた OGR ライブラリでサポートされているベクタフォーマット、具体的には GeoPackage、ESRI Shapefile、MapInfo、SDTS、GML、などその他多数。詳しくは **ベクタデータの操作** のセクションを参照してください。
- インストールされた GDAL (Geospatial Data Abstraction Library) でサポートされているラスタフォーマットと画像フォーマット、具体的には GeoTIFF、ERDAS IMG、ArcInfo ASCII GRID、JPEG、PNG、などその他多数。詳しくは **ラスタデータの操作** のセクションを参照してください。
- メッシュデータ（TIN とレギュラーグリッドがサポートされています）。詳しくは **メッシュデータの操作** を参照してください。
- ベクタタイル
- GRASS データベース（location/mapset）から提供される GRASS ラスタデータとベクタデータ。 *GRASS GIS* **の統合** を参照して下さい.
- OGC Web サービスとして提供されているオンライン空間データ、つまり WMS、WMTS、WCS、WFS、WFS-T。 *OGC / ISO* **プロトコルで作業する** を参照して下さい。

  QGIS 認証基盤は Web サービスその他のリソースにおけるユーザーとパスワード、証明書とキーの管理の助けとなります。
- スプレッドシート（ODS / XLSX）

時系列データ

4.2 データの検索と地図の作成

親しみやすい GUI を通して、地図の作成と、空間データのインタラクティブな検索ができます。以下のような多くの便利なツールが GUI で利用可能です。

- QGIS ブラウザ
- オンザフライ再投影
- 2D と 3D 地図レンダリング
- DB マネージャ
- 印刷レイアウト
- レポート
- 全体図パネル
- 空間ブックマーク
- 注記ツール
- 地物情報表示/選択
- 属性の編集/表示/検索
- データ定義の地物ラベリング
- データ定義のベクタおよびラスタシンボロジツール
- グリッドレイヤを使った地図帳の構成
- 地図に表示する北向き矢印、スケールバー、著作権表示
- プロジェクトの保存と読み込みのサポート

4.3 データの作成、編集、管理とエクスポート

ベクタレイヤやラスタレイヤを作成、編集、管理し、さまざまな形式でエクスポートできます。QGIS は以下の機能を提供しています。

- ベクタデジタイジングツール
- 複数のファイルフォーマットや GRASS ベクタレイヤを作成し編集する機能
- ベクタや画像をジオコーディングするためのジオリファレンサツール
- GPX 形式のインポートやエクスポート、その他の GPS 形式を GPX に変換したり GPS ユニット（Linux では usb: は GPS デバイスのリストに追加される）から直接ダウンロード/アップロードするための GPS ツール
- OpenStreetMap データの可視化と編集のサポート
- DB マネージャプラグインによりファイルから空間データベースを作る機能

- 空間データベーステーブルのより進んだ扱い
- ベクタ属性テーブルを管理するツール
- スクリーンショットをジオリファレンスされた画像として保存するオプション
- スタイルを出力する拡張された性能をもつ DXF 出力ツールや、CAD のような機能が備わったプラグイン

4.4 データの解析

空間データベースやその他の OGR がサポートするフォーマットに対して空間データ解析を行うことができます。QGIS は現在、ベクタ解析、ラスタ解析、サンプリング、ジオプロセシング、ジオメトリ、データベース管理の各ツールを提供しています。また、400 以上のモジュールからなる完全な GRASS の機能を含む統合 GRASS ツールも使用することができます（ *GRASS GIS* **の統合** のセクションを参照）。また、プロセシングプラグインを使用して作業することもできます。これは強力な地理空間解析フレームワークを提供し、ネイティブのアルゴリズムや GDAL、SAGA、GRASS、OTB、R やその他のサードパーティのアルゴリズムを QGIS から呼び出すことができます（ **はじめに** のセクションを参照）。すべての解析関数はバックグラウンドで実行されるため、処理の完了を待たずに作業を続行することができます。

グラフィカルモデラーによって、直感的なグラフィカルな環境のもとで、複数の機能を組み合わせたり繋げたりして完全なワークフローを作り上げることが可能です。

4.5 インターネットでの地図の公開

QGIS は WMS、WMTS、WMS-C、WFS、OAPIF、WFS-T クライアントとして使用でき（ *OGC / ISO* **プロトコルで作業する** を参照）、QGIS Server（ QGIS-Server-manual を参照）を使用して、WMS、WCS、WFS、OAPIF プロトコルによるデータを Web サーバーを使ってインターネットで公開することができます。

4.6 プラグインによる QGIS 機能の拡張

QGIS は、拡張可能なプラグインアーキテクチャとプラグイン作成用のライブラリによって、あなたの特別なニーズにも応えることができます。C++や Python を使って、新たなアプリケーションを作ることさえも可能です！

4.6.1 コアプラグイン

コアプラグインに含まれているものは以下の通りです。

1. DB マネージャ（レイヤの交換・編集・表示とテーブルとデータベースの相互変換、SQL クエリの実行）
2. ジオメトリチェッカー（ジオメトリのエラーをチェックします）
3. GDAL のジオリファレンサ （GDAL を利用して、ラスタに投影情報を付加します）

4. GPS ツール （GPS データのロードとインポート）
5. GRASS（GRASS GIS の統合）
6. メタサーチカタログクライアント（Web 用の OGC カタログサービス （CSW）規格をサポートするメタデータカタログサービスの操作）
7. オフライン編集（データベースのオフライン編集と同期）
8. プロセシング（QGIS 用の空間データプロセシングフレームワーク）
9. トポロジチェッカー（ベクタレイヤ内のトポロジーエラーを検出する）

4.6.2 外部 Python プラグイン

QGIS が公開する、コミュニティによって提供される外部 Python プラグインの数は増え続けています。これらのプラグインは公式のプラグインリポジトリにあり、Python プラグインインストーラを使用して簡単にインストールできます。 **プラグインダイアログ** のセクションを参照してください。

4.7 Python コンソール

スクリプト実行には統合された Python コンソールを利用することが可能です。コンソールは **プラグイン** *Python* **コンソール** メニューから開くことができます。コンソールは非モーダルユーティリティウィンドウとして開きます。QGIS 環境をインタラクティブに利用するために `QgisInterface` クラスのインスタンスである `qgis.utils.iface` という変数が利用できます。このインターフェイスでは地図キャンバス、メニュー、ツールバー及び QGIS アプリケーションのその他の部分へのアクセスを提供します。スクリプトを作成して、その後 QGIS ウィンドウにドラッグアンドドロップすると自動的に実行できます。

Python コンソールおよび QGIS のプラグインやアプリケーションのプログラミングに関する詳細については PyQGIS-Developer-Cookbook を参照して下さい。

4.8 既知の問題

4.8.1 ファイル数の制限

もし大きな QGIS プロジェクトを開いていて、多くのレイヤが正常だけどいくつかのレイヤがおかしい場合、おそらくこの問題に遭遇しています。Linux（そして他の OS でも同様）では、あるプロセスが開けるファイルの数に制限があります。リソースの制限はプロセスごとであり、これは小プロセスにも継承されます。シェル組み込みの `ulimit` コマンドを使うと、現在のシェルプロセスについてその制限を変更できます。新しい制限は、すべての子プロセスに継承されます。

以下のように入力すると、すべての現在の ulimit 情報を見ることができます :

```
$ ulimit -aS
```

コンソール上で以下のコマンドを使用すると、現在許容されているプロセスあたりの開かれたファイルの数を見ることができます:

```
$ ulimit -Sn
```

既存のセッション の制限を変更したい場合は、次のような操作で可能です :

```
$ ulimit -Sn #number_of_allowed_open_files
$ ulimit -Sn
$ qgis
```

代わりに、新しい `prlimit` ユーティリティを使用することもできます。詳細については、 https://manpages.ubuntu.com/manpages/latest/man1/prlimit.1.html を参照してください。

問題を永続的に解決するためには

ほとんどの Linux システムでは、リソースの制限はログイン時に `pam_limits` モジュールによって設定されます。この制限は、`/etc/security/limits.conf` または `/etc/security/limits.d/*.conf` ファイル内に含まれている設定に従います。root 権限があれば（sudo も可）これらのファイルを編集することができるはずですが、 変更が有効になるには、もう一度ログインする必要があります。

更なる情報:

https://www.cyberciti.biz/faq/linux-increase-the-maximum-number-of-open-files/ https://linuxaria.com/article/open-files-in-linux

第5章　最初のステップ

この章では、QGIS のインストール、QGIS サンプルデータのダウンロードと、ラスタとベクタデータを表示する簡単な最初のセッションの実行について簡単に説明します。

5.1 QGIS のインストール

QGIS プロジェクトでは、QGIS をインストールするためのさまざまな方法をお使いのプラットフォームに応じて提供しています。

5.1.1 バイナリからインストールする

MS Windows および macOS 用では、標準インストーラが提供されています。 GNU / Linux 各種向けには、バイナリパッケージ（rpm および deb）またはソフトウェアリポジトリが提供されています。

お使いのオペレーティングシステム向けの更なる情報と手順については https://download.qgis.org を参照して下さい。

5.1.2 ソースからのインストール

QGIS をソースからビルドする必要がある場合には、インストール手順を参照してください。これは `INSTALL` というファイル名で QGIS のソースコードとともに配布されています。オンラインで https://github.com/qgis/QGIS/blob/release-3_28/INSTALL.md でも参照できます。

開発中のバージョンではない、特定のリリースをビルドする場合は、上記のリンクの `master` をリリースブランチ（通常は `release-X_Y` という形式）に置き換えてください（インストール手順は異なる場合があります）。

5.1.3 外部メディアへのインストール

QGIS を（すべてのプラグインと設定を含めて）フラッシュドライブにインストールすることが可能です。これは、*--profiles-path* オプションを定義してデフォルトの *user profile* パスを上書きし、**QSettings** ディレクトリを使用するように設定することで実現できます。 詳しい情報は、 **システムの設定** セクションを参照して下さい。

5.1.4 サンプルデータのダウンロード

このユーザーガイドには、QGIS サンプルデータセット（`Alaska dataset` とも呼びます）に基づく例があります。サンプルデータは https://github.com/qgis/QGIS-Sample-Data/archive/master.zip からダウンロードし、このアーカイブをコンピュータの好きな場所に展開してください。

Alaska dataset には、このユーザーガイドのサンプルとスクリーンショットで使用されているすべての GIS データが含まれています。また、小さな GRASS データベースも含まれています。QGIS サンプルデータセットの投影法は、フィートを単位とする Alaska Albers Equal Area 図法です。EPSG コードは 2964 です。

```
PROJCS["Albers Equal Area",
GEOGCS["NAD27",
DATUM["North_American_Datum_1927",
SPHEROID["Clarke 1866",6378206.4,294.978698213898,
AUTHORITY["EPSG","7008"]],
TOWGS84[-3,142,183,0,0,0,0],
AUTHORITY["EPSG","6267"]],
PRIMEM["Greenwich",0,
AUTHORITY["EPSG","8901"]],
UNIT["degree",0.0174532925199433,
AUTHORITY["EPSG","9108"]],
AUTHORITY["EPSG","4267"]],
PROJECTION["Albers_Conic_Equal_Area"],
PARAMETER["standard_parallel_1",55],
PARAMETER["standard_parallel_2",65],
PARAMETER["latitude_of_center",50],
PARAMETER["longitude_of_center",-154],
PARAMETER["false_easting",0],
PARAMETER["false_northing",0],
UNIT["us_survey_feet",0.3048006096012192]]
```

GRASS 用のグラフィカルフロントエンドとして QGIS を使用したい場合には、GRASS GIS の公式ウェブサイト https://grass.osgeo.org/download/data/ でロケーションのサンプルのセレクション（例えば、Spearfish や South Dakota）を見つけることができます。

5.2 QGIS の起動と終了

QGIS は、他のアプリケーションと同じように、次のような方法で起動することができます。

- ではアプリケーションメニューを、 ではスタートメニューを、 では Dock を使って起動する
- アプリケーションフォルダまたはデスクトップショートカットのアイコンをダブルクリックする
- 既存の QGIS プロジェクトファイル（拡張子は `.qgz` または `.qgs` ）をダブルクリックする（この操作では、プロジェクトも同時に開かれることに注意してください）
- コマンドプロンプトで `qgis` とタイプする（QGIS のインストール場所が PATH に追加されているか、インストールフォルダに移動していることが前提となります）

QGIS を終了させる場合は以下の方法を使用します。

- メニューから **プロジェクト** *QGIS* **を終了** を選択する、またはショートカット Ctrl キー+Q を使用する
- メニューから *QGIS* *QGIS* **を終了** を選択する、またはショートカット Cmd+Q を使用する
- アプリケーションのメインインターフェースの右上角にある赤いバツ印を使う

5.3 サンプルセッション：ラスタレイヤとベクタレイヤを読み込む

さて、 *QGIS* **をインストール** して **サンプルデータセット** が利用できるようになりましたので、初めてのサンプルセッションを実行してみましょう。この例ではラスタレイヤとベクタレイヤを表示します。以下のデータを使用します。

- landcover ラスタレイヤ (qgis_sample_data/raster/landcover.img)
- lakes ベクタレイヤ (qgis_sample_data/gml/lakes.gml)

qgis_sample_data はデータセットを展開したフォルダのパスを意味します。

1. *QGIS* **の起動と終了** で説明されている方法で QGIS を起動します。
2. 今回扱うデータは Albers Equal Area 図法で作成されています。このため、プロジェクトの CRS を適切に設定しましょう:
    1. QGIS インターフェースの右下にある CRS を選択 ボタンをクリックします。プロジェクトのプロパティダイアログが開き、 *CRS* タブがアクティブになります。
    2. **フィルタ** のテキストボックスに 2964 と入力します。
    3. NAD27 / Alaska Albers という CRS 名の行を選択します。

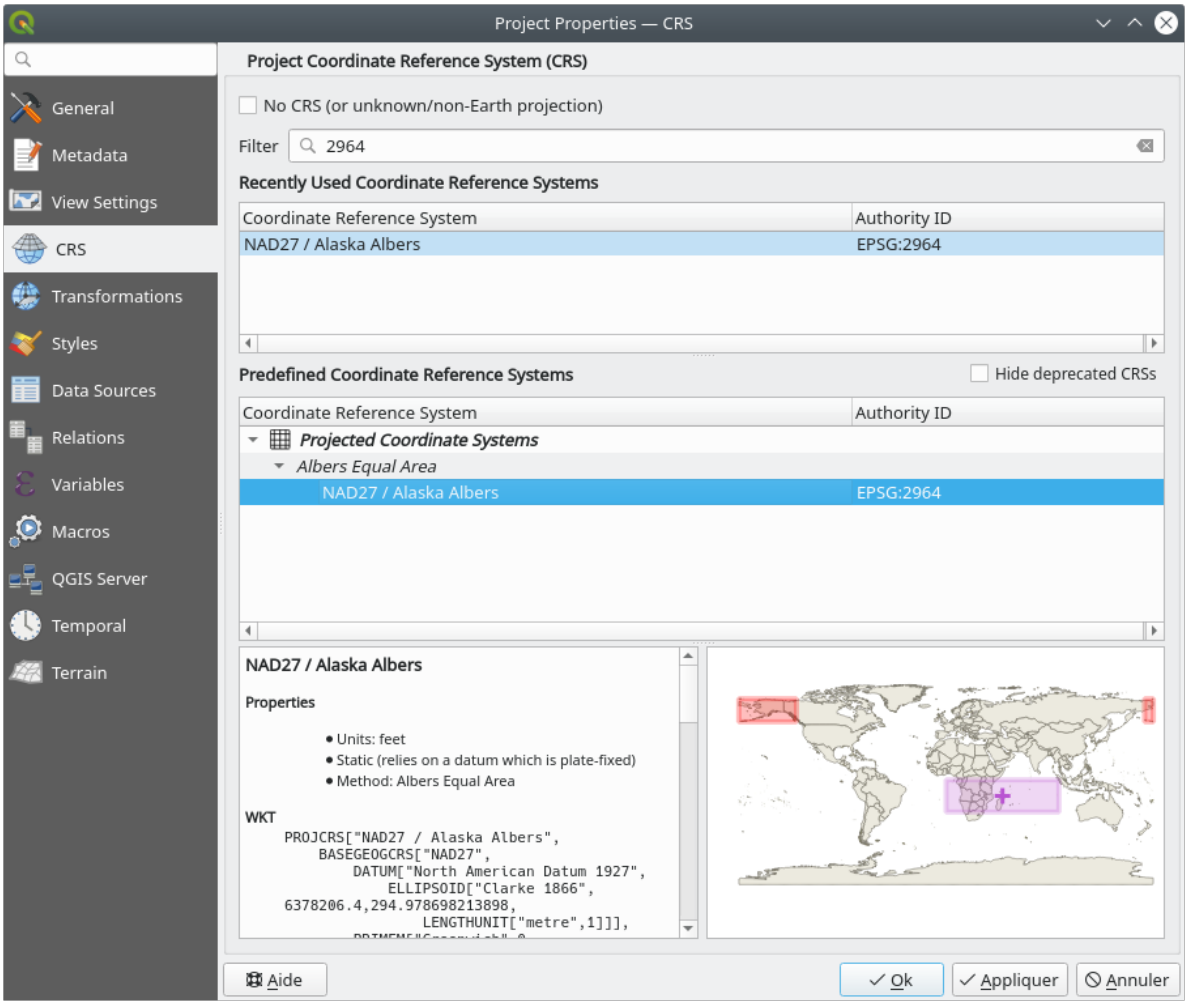


図 5.1: データの座標参照系を選択する

4. *OK* を押します

注釈: "概算 transform が使用されました" というメッセージが表示されるかもしれませんが、今のところは無視するか閉じてしまってかまいません。

---

3. QGIS にファイルを読み込みます:

   1. 

アイコンをクリックします。データソースマネージャがブラウザモードで開きます。

   2. `qgis_sample_data/raster/` フォルダに移動します。

   3. ERDAS IMG ファイル  *landcover.img* を選択してダブルクリックします。データソースマネージャウィンドウは開いたままで、landcover レイヤがバックグラウンドで追加されます。

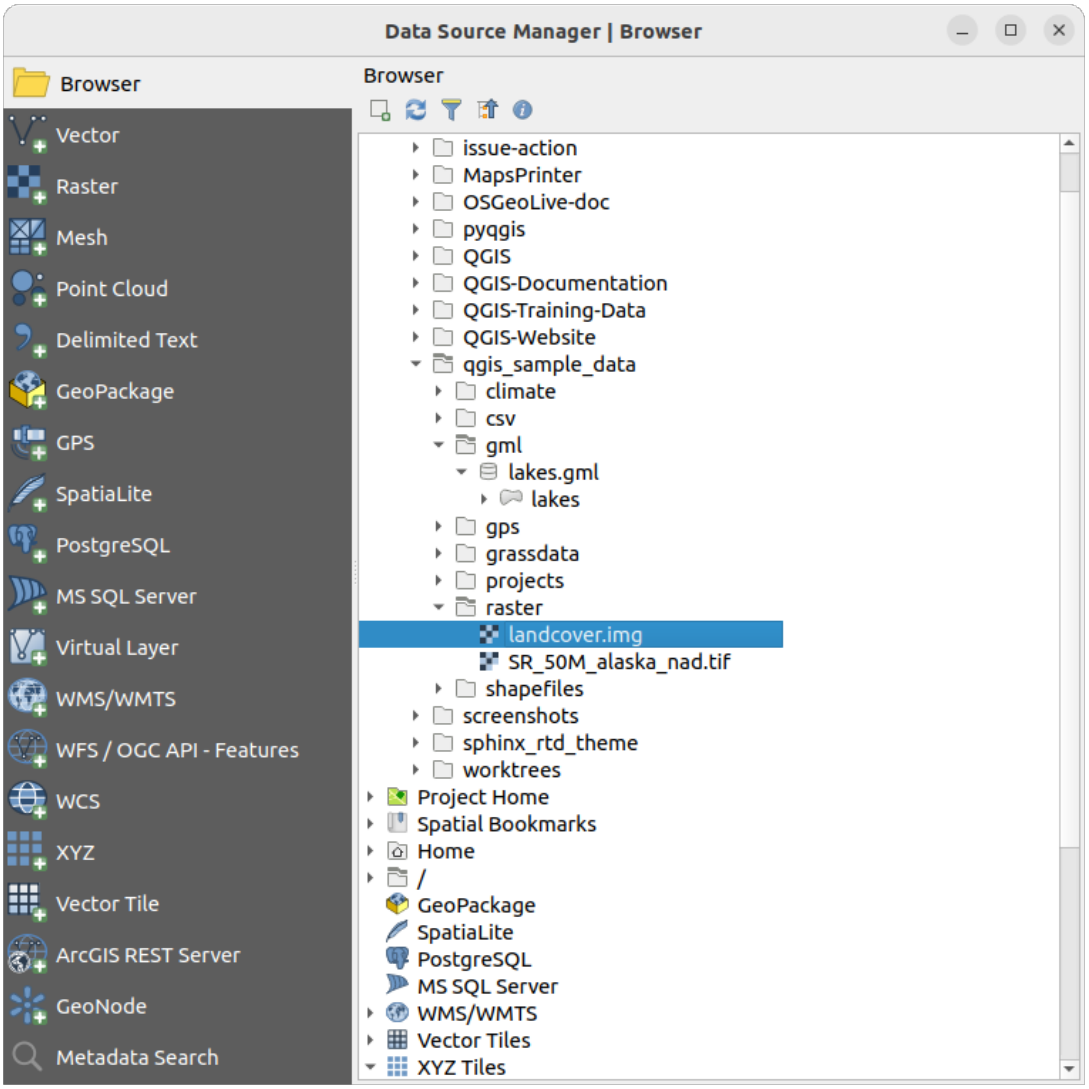


図 5.2: 新しい QGIS プロジェクトにデータを追加する

4. lakes データを読み込むには、`qgis_sample_data/gml/` フォルダに移動し、*lakes.gml* ファイルを QGIS のメインダイアログにドラッグ＆ドロップします（または、上で述べたようにファイルをダブルクリックします）。

5. **追加するアイテムを選択** ダイアログが開き、ファイルをスキャンします。これは、`.gml` ファイル形式が一度に複数のレイヤを保存できるためです。

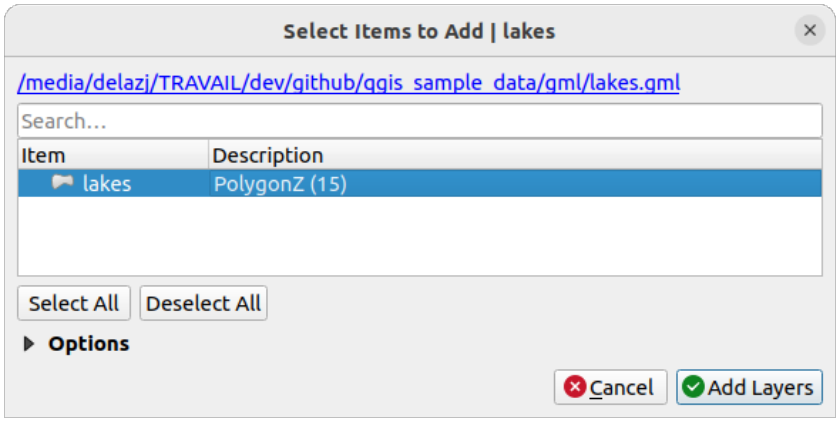


図 5.3: ファイル内のレイヤの選択

6. 今回は、単一の *lakes* があります。これを選択し、 **レイヤを追加** ボタンを押します。
7. **レイヤ** パネルにレイヤが追加されます。

4. データソースマネージャウィンドウを閉じます。

レイヤ パネルにおいて、 *lakes* レイヤの横に レイヤに CRS が設定されていません と表示されているのに気づくでしょう。これを解決しましょう。

1. アイコンをクリックします。 *CRS* **セレクタ** ダイアログが開きます。
2. 上で行ったのと同様に、 *NAD27 / Alaska Albers* の CRS エントリを検索して選択します。
3. *OK* をクリックします。

これでプロジェクトで 2 つのレイヤを使用できるようになりました。レイヤは QGIS によってランダムな配色がされています。では、 lakes レイヤをカスタマイズしてみましょう。

1. **ナビゲーション** ツールバーで 拡大 ツールを選択します。
2. 湖がある場所にズームします。
3. `lakes` レイヤをレイヤパネルでダブルクリックして **レイヤプロパティ** ダイアログを開きます。
4. 湖の色を変えるには、次のように操作します：
    1. **シンボロジ** タブをクリックします。
    2. 塗りつぶし色として青を選択します。

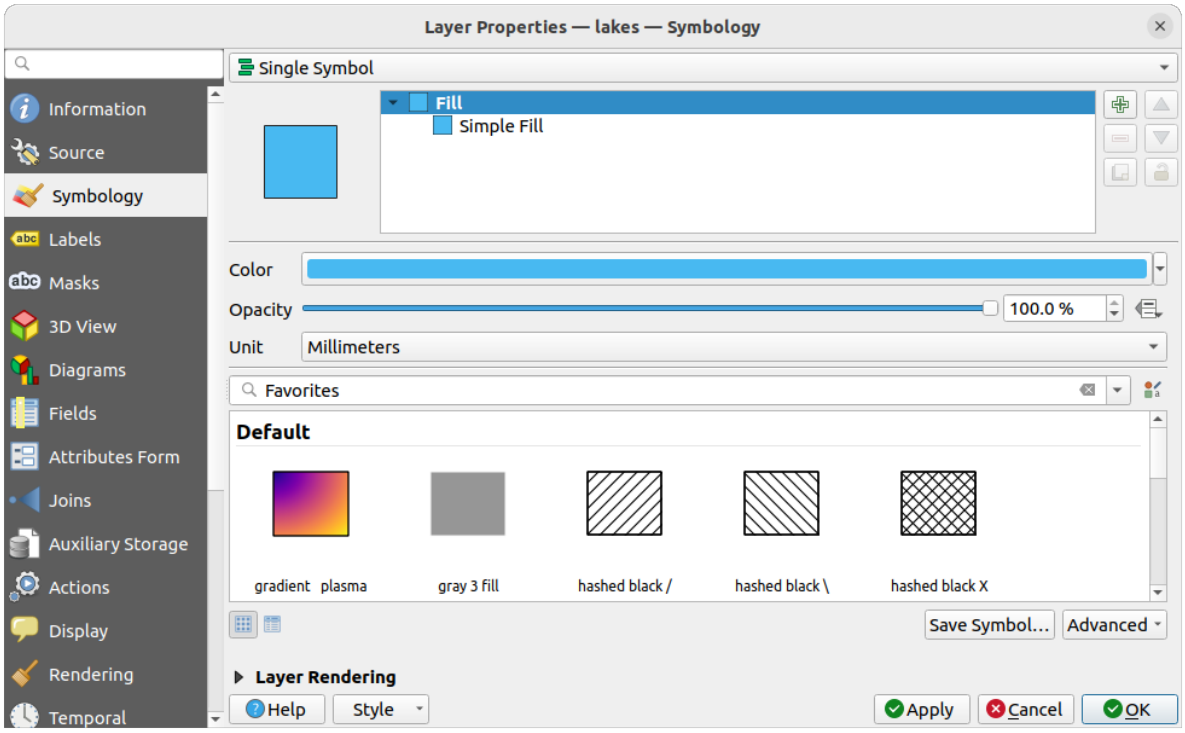


図 5.4: 湖の色を選択する

3. *OK* を押します。マップキャンバスでは、湖が青色で表示されます。

5. 湖の名前を表示するには、次のように操作します：

   1. lakes レイヤで **レイヤプロパティ** ダイアログを再度開きます。
   2. **ラベル** タブをクリックします。
   3. ドロップダウンメニューで **単一定義** *(single)* を選択すると、ラベリングが有効になります。
   4. **値** リストから NAMES フィールドを選びます。

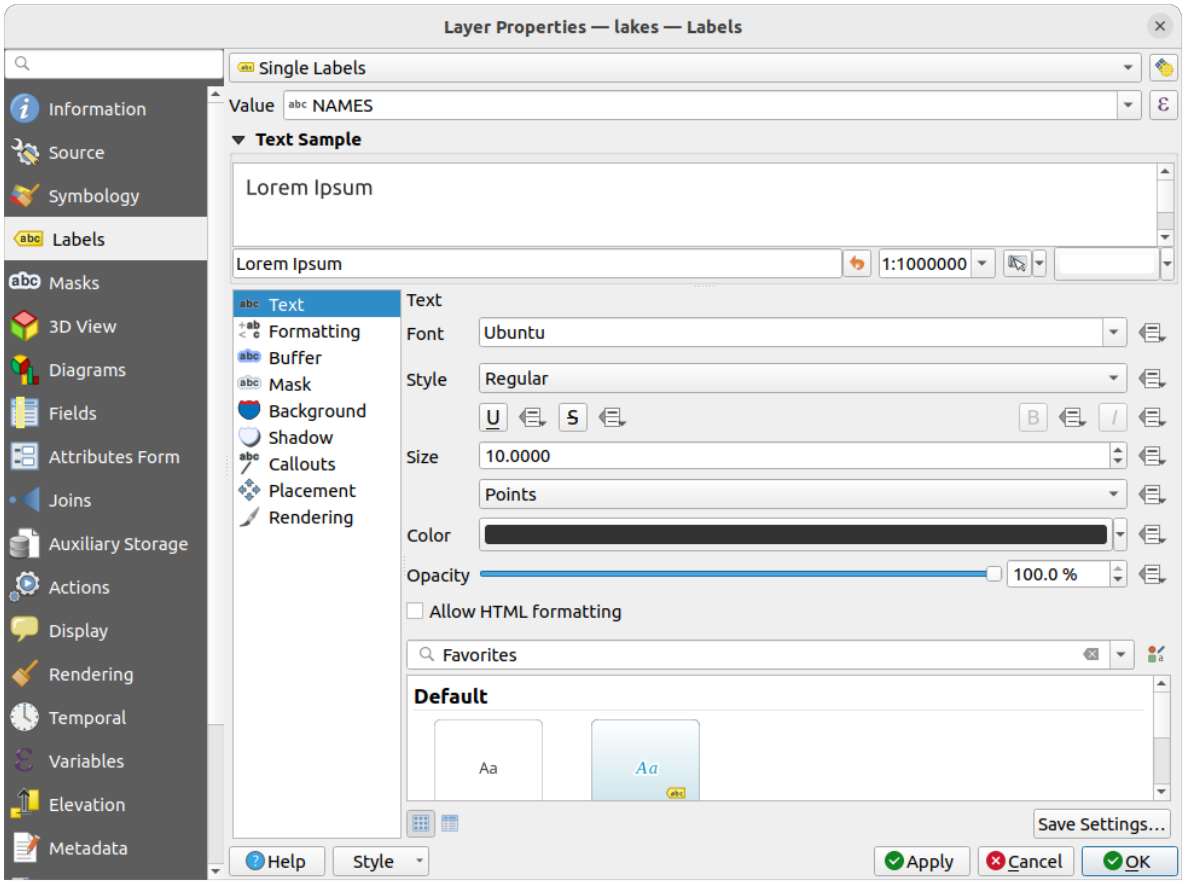


図 5.5: 湖の名前を表示する

5. **適用** を押します。すると名前が湖の領域の上に表示されます。

6. ラベルに白いバッファを追加してラベルを読みやすくするには、次のように操作します：

   1. 左のリストで **バッファ** タブをクリックします。
   2. **テキストバッファを描画** にチェックを入れます。
   3. サイズを 3 にします。
   4. **適用** をクリックします。
   5. 出来栄えを確認して、必要なら値を変更します。
   6. 最後に *OK* をクリックして **レイヤプロパティ** ダイアログを閉じ、変更を適用します。

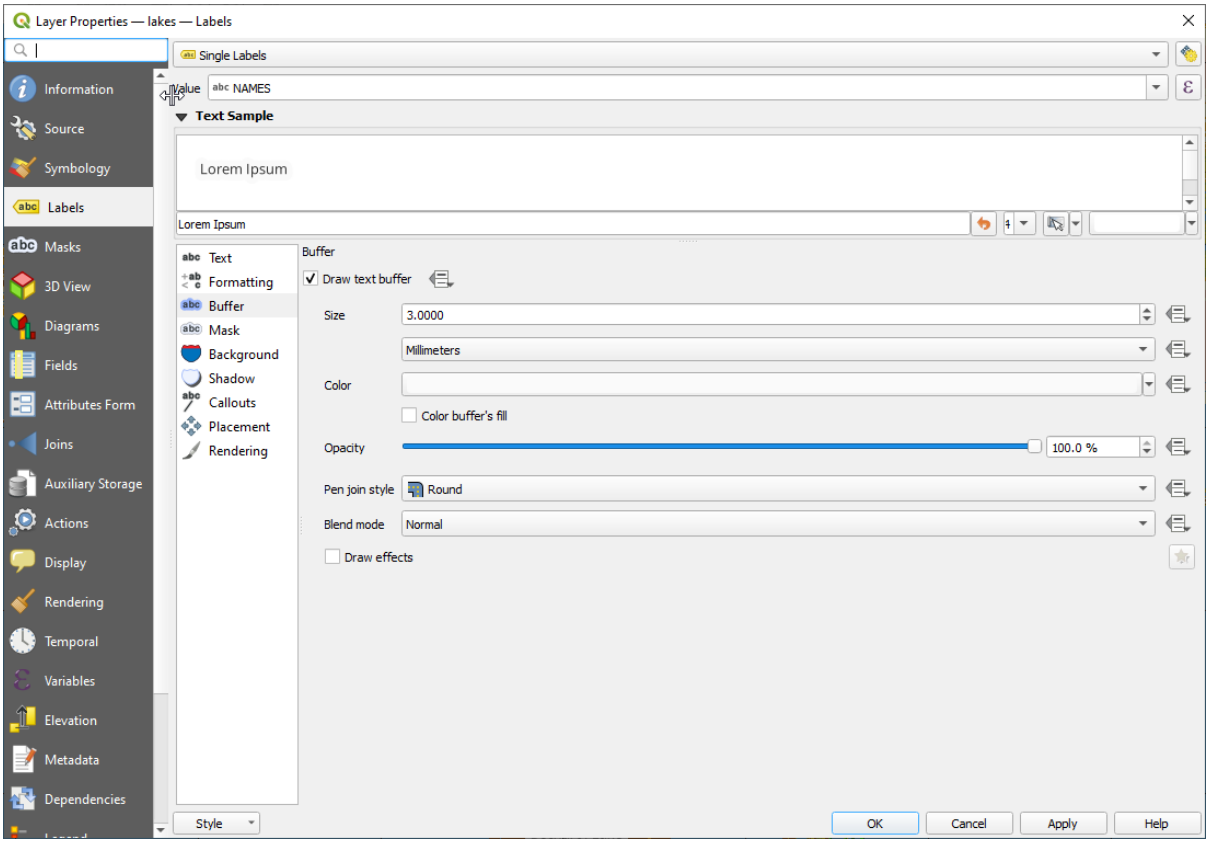


図 5.6: ラベルの周囲にバッファを表示する

今度は、地図の見栄えを整えるためにいくつか整飾を加えて、QGIS からエクスポートしてみましょう。

1. メニューから **ビュー** **地図整飾** **スケールバー** を選択します。
2. 開いたダイアログで、 **スケールバーを有効にする** にチェックを入れます。
3. 好みにあわせてダイアログのオプションを設定します。
4. **適用** を押します。
5. 同様に地図整飾メニューから、さらに方位記号、著作権ラベルなどのアイテムを、属性を調整して地図キャンバスに追加できます。
6. メニューから **プロジェクト** **インポートとエクスポート** **地図を画像にエクスポート...** を選びます。
7. 開いたダイアログで **保存** を押します
8. ファイルの保存場所とフォーマットを選択し、再度 **保存** を押して確定します。
9. 変更を `.qgz` プロジェクトファイルに保存する場合は、 **プロジェクト** **保存...** を選択します。

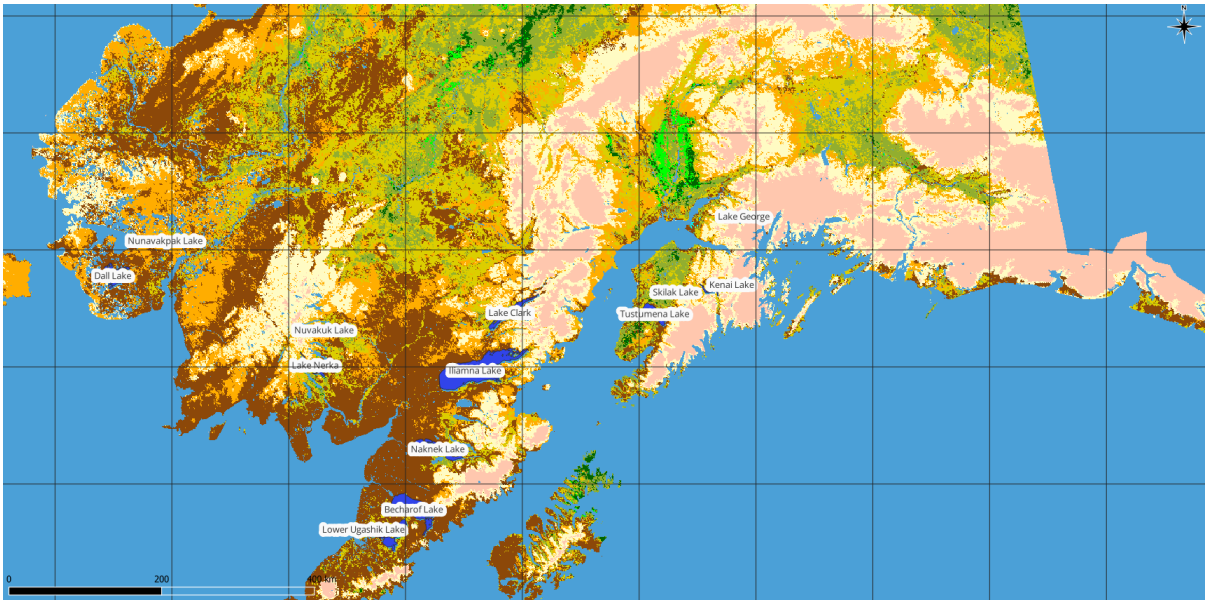


図 5.7: 整飾されたエクスポート地図を表示する

これでおしまいです。QGIS でラスタレイヤやベクタレイヤを可視化し、設定を行った上で、他のソフトウェアで使用できる画像フォーマットで地図を作成するのがいかに簡単であるか理解できたと思います。引き続き、QGIS で使える機能や特徴、設定、およびその使用方法についてさらに学んでいきましょう。

注釈: QGIS をステップバイステップの演習で勉強したい場合は トレーニングマニュアル を参照して下さい。

第6章　プロジェクトファイルでの作業

6.1 QGIS プロジェクトの紹介

QGIS セッションの状態は「プロジェクト」と呼ばれます。QGIS が動作するのは一度に 1 つのプロジェクトです。新しいプロジェクトの設定には、プロジェクトに固有のものや、アプリケーション全体のデフォルトで決められているものがあります（ **オプション** 参照）。QGIS では、メニューオプションの **プロジェクト** **保存** または **プロジェクト** **名前をつけて保存**... を利用することで、ワークスペースの状態を *QGIS* **プロジェクトファイル** に保存できます。

注釈: プロジェクトの内容に変更があった場合、タイトルバーには * 記号が表示され、デフォルトでは QGIS は変更を保存するかどうかを確認します。この動作は、 **設定** **オプション** **一般情報** にある **必要なときにプロジェクトおよびデータソースの変更を保存するか尋ねる** の設定によって制御されます。

既存のプロジェクトを QGIS に読み込むには、ブラウザパネルを利用するか、 **プロジェクト** **開く**... 、 **プロジェクト** **テンプレートから新規作成** 、または **プロジェクト** **最近使用したプロジェクト** を使用します。

起動時には、 **プロジェクトのテンプレート** と、（最大 10 個の）スクリーンショット、名前、ファイルパスの情報が含まれる **最近使用したプロジェクト** のリストが表示されます。 **最近使用したプロジェクト** のリストは、最近使ったプロジェクトへのアクセスに便利です。最近使ったプロジェクトやプロジェクトのテンプレートを開くには、エントリをダブルクリックします。エントリを右クリックすると、 **リストに登録** 、 **ディレクトリを開く**... 、 **リストから削除** の操作ができます。レイヤを追加することでも、新規プロジェクトを自動的に作成することができます。リストが消え、マップキャンバスが表示されます。

セッションをクリアして新たなセッションを開始したい場合は、 **プロジェクト** **新規** を実行します。このとき、プロジェクトを開いてからや最後に保存されて以降に変更があった場合には、現在のプロジェクトを保存するように促されます。

新規プロジェクトを開くと、保存するまではタイトルバーには 無題のプロジェクト と表示されます。

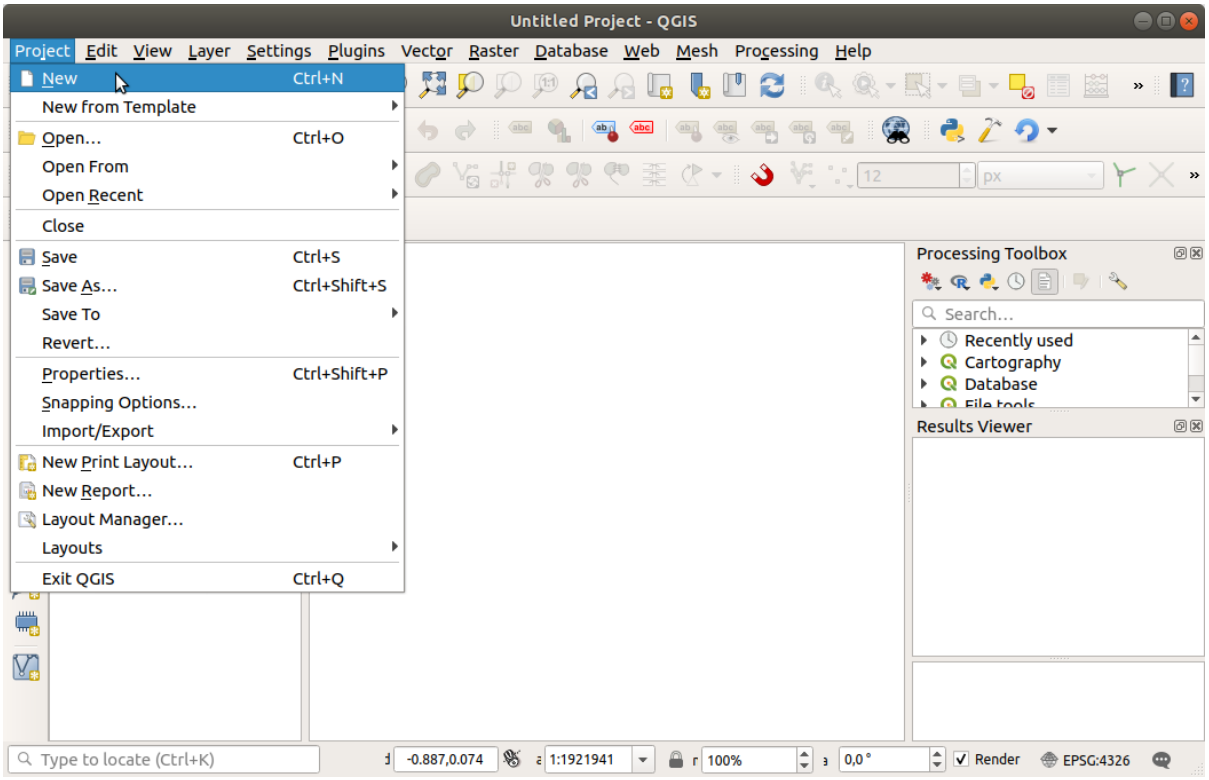


図 6.1: QGIS で新しいプロジェクトを開始する

以下の情報がプロジェクトファイルに保存されます：

- 追加されたレイヤ
- どのレイヤをクエリできるか
- シンボルとスタイルを含むレイヤのプロパティ
- レイヤノート
- 2D と 3D のマップビュー
- 各マップビューの投影法
- 各マップの最後に表示された範囲
- 印刷レイアウト
- 印刷レイアウト要素とその設定
- 印刷レイアウトの地図帳の設定
- デジタイズの設定
- テーブルのリレーション
- プロジェクトのマクロ
- プロジェクトのデフォルトのスタイル
- プラグイン設定

- QGIS サーバーの設定（プロジェクトプロパティの OWS 設定タブの設定）
- DB マネージャに格納されたクエリ

プロジェクトファイルは XML 形式で保存されます（ *QGS/QGZ - QGIS* **プロジェクトファイル形式** 参照）。これは、知識があれば QGIS の外部でファイルを編集することが可能であることを意味します。プロジェクトファイルの形式は何度か更新されています。QGIS の古いバージョンのプロジェクトファイルは、今後は正常に動作しない可能性があります。

注釈: デフォルトでは、QGIS は開いたプロジェクトファイルとプログラムのバージョンが異なると警告します。この動作は **設定** **オプション** の **一般情報** タブ（ **古いバージョンの** *QGIS* **で保存されたプロジェクトファイルを開く時に警告する** ）で制御できます。

QGIS で `.qgs` プロジェクトファイルを保存すると、同じディレクトリに `.qgs~` という拡張子のバックアップファイルが作成されます。

QGIS プロジェクトの拡張子は `.qgs` ですが、QGIS から保存する場合、デフォルトでは `.qgz` という拡張子の圧縮形式で保存されます。`.qgs` ファイルは `.qgz` ファイル（zip アーカイブ）の中に、関連する SQLite データベース（`.qgd`）と一緒に *auxiliary data* として埋め込まれます。これらのファイルは `.qgz` ファイルを展開することで見ることができます。

注釈: 補助データを埋め込めるので、 **補助テーブルプロパティ** の仕組みは zip 形式のプロジェクトの利便性を高めます。

プロジェクトは、以下のプロジェクトメニュー項目を使用して、PostgreSQL、GeoPackage、または Oracle データベースに保存/読み込みすることもできます。

- **プロジェクト** **開く**
- **プロジェクト** **保存**

どちらのメニュー項目にもサブメニューがあり、追加プロジェクト用のストレージ実装（PostgreSQL、GeoPackage、Oracle）のリストがあります。アクションをクリックすると、GeoPackage の接続とプロジェクト、PostgreSQL の接続とスキーマとプロジェクト、Oracle の接続とオーナーとプロジェクトを選択するためのダイアログが開きます。

GeoPackage、PostgreSQL、Oracle に保存されているプロジェクトは、QGIS のブラウザパネルからダブルクリックするか、マップキャンバスにドラッグするかして読み込むこともできます。

6.2 壊れたファイルパスの取り扱い

プロジェクトを開くときに、利用できないサービス/データベース、またはファイル名の変更や移動によって、QGIS が一部のデータソースに到達できないことがあります。この場合、QGIS は **利用できないレイヤを処理** ダイアログを開き、見つからなかったレイヤを列挙します。以下の操作ができます。

- **データソース** フィールドをダブルクリックし、各レイヤのパスを修正して **変更の適用** ボタンをクリックします。
- 行を選択し、 **ブラウズ** ボタンを押して正しい場所を指定し、 **変更の適用** ボタンをクリックします。
- **自動検索** ボタンを押し、フォルダを検索して、全てもしくは選択したファイルの壊れているパスの自動修正を試みます。検索には時間がかかることがあるため注意してください。見つかったら、 **変更の適用** をクリックしてください。
- **利用できないレイヤを保持する** ボタンをクリックすると、メッセージを無視して、パスが壊れた状態のままでプロジェクトを開きます。この場合、レイヤは **レイヤ** パネルに表示されますが、 **レイヤ** パネルの横にある  利用できないレイヤ! アイコンまたはレイヤのコンテキストメニューの **データソースの変更**... を使用してパスを修正するまではデータが空になります。

  データソースの変更... ツールを使用して、あるレイヤのパスが修正されると、QGIS は他の壊れたパス全てをスキャンし、同じ壊れたファイルパスの自動修正を試みます。
- プロジェクトから  **利用できないレイヤを削除** します。

--skipbadlayers オプションを使用してコマンドラインから QGIS を起動すると、起動時の **利用できないレイヤを処理** ダイアログをスキップすることができます。

6.3 出力を作成する

QGIS セッションから出力を作成するには、いくつかの方法があります。 *QGIS* **プロジェクトの紹介** でプロジェクトファイルとして保存することについては既に説明しました。出力ファイルを作成する他の方法は：

- 画像の作成： **プロジェクト** **インポート/エクスポート** **|地図を画像として保存| 地図を画像にエクスポート**... を実行すると、キャンバスに表示されている地図を独自のスケール、解像度、サイズで画像形式 (PNG, JPG, TIFF...) で出力します。チェックボックス| **地図参照情報を追加** *(embedded or via world file)* **`を有効にしてください。詳しくは** *:ref:`exportingmapcanvas* を参照してください。
- PDF ファイルへの出力： **プロジェクト** **インポートとエクスポート** **地図を** *PDF* **へエクスポート**... を使うと、カスタムのスケール、解像度といくつかの詳細設定（簡素化、ジオリファレンス等）を指定してマップキャンバスを PDF に出力することができます。詳細は **マップビューのエクスポート** を参照して下さい。
- DXF ファイルに出力する： **プロジェクト** **インポートとエクスポート** **プロジェクトを** *DXF* **にエクスポート**... を選ぶとダイアログが開き、「シンボロジーモード」、「シンボルの縮尺」 、DXF に出力するベクタレイヤを定義できます。「シンボロジーモード」によって、QGIS のオリジナルシンボルを忠実に出力できます（ **新しい** *DXF* **ファイルを作成する** セクションを参照）。

- 地図をデザインする： **プロジェクト**  **新規印刷レイアウト**... を選択するとダイアログが開き、現在のマップキャンバスをレイアウトして印刷することができます（ **地図のレイアウト** セクションを参照)。

第7章　QGISのユーザインタフェース

QGIS のグラフィカルユーザインタフェース（GUI） を以下に図で示します。黄色い丸の 1 から 5 の番号は QGIS GUI の重要な構成要素を表しています。それらについて以下で説明します。

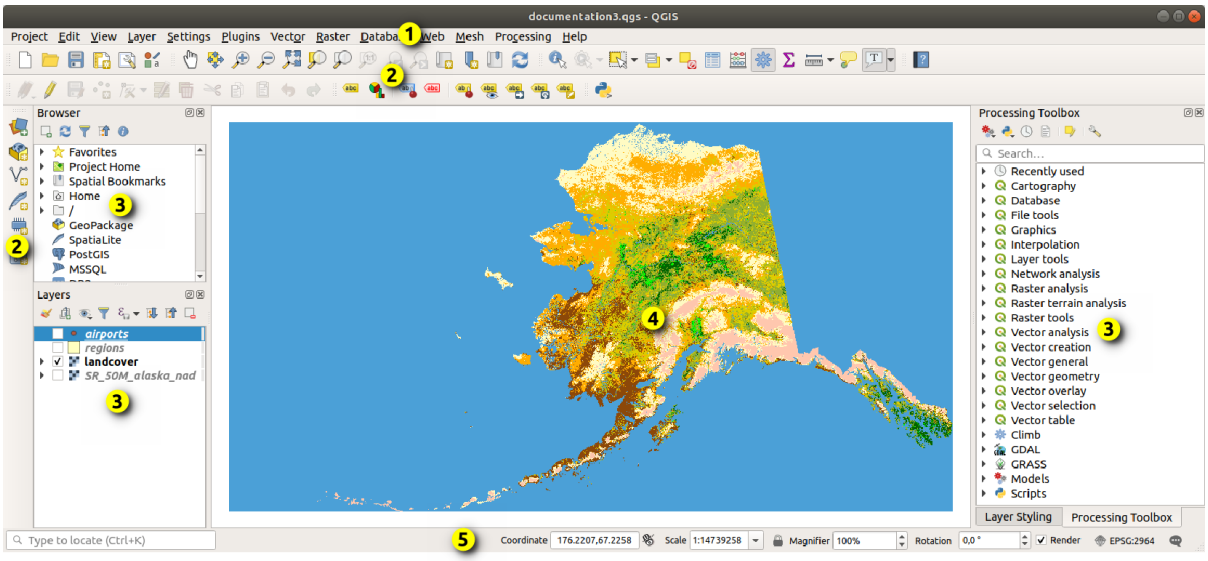


図 7.1: Alaska サンプルデータを開いた QGIS GUI

注釈: ウィンドウの装飾（タイトルバー等）は、利用しているオペレーティングシステムやウィンドウマネージャによって表示が異なることがあります。

QGIS のメイン GUI（ 図 7.1 ） は５つのコンポーネントで構成されています。コンポーネントのタイプは以下のとおりです：

1. **メニューバー**
2. **ツールバー**
3. **パネル**
4. **マップビュー**
5. **ステータスバー**

下にスクロールすると、これらの詳細な説明があります。

7.1 メニューバー

メニューバーでは、標準的な階層型メニューを使用して QGIS の機能にアクセスすることができます。メニュー、メニューのオプション、関連するアイコン、およびキーボードショートカットについて以下で説明します。キーボード ショートカットは再設定することができます（ **設定** **キーボードショートカット** ）。

ほとんどのメニューオプションには対応するツールがあり、その逆も同様です。ただし、メニューはツールバーとまったく同じように構成されてはいません。ツールバーのメニューオプションの場所を以下の表に示します。プラグインはメニューに新しいオプションを追加する場合があります。ツールとツールバーの詳細については、**ツールバー** を 参照してください。

注釈: QGIS はクロスプラットフォームのアプリケーションです。ツールは通常はすべてのプラットフォームで使用できますが、オペレーティングシステムによっては異なるメニュー位置に配置される場合があります。以下のリストは、既知のバリエーションを含む、最も一般的な場所を示しています。

7.1.1 プロジェクト

プロジェクト メニューには、**プロジェクトファイル** へのアクセスポイントと終了ポイントがあります。以下のツールを提供します:

- **新規** プロジェクトファイルを最初から作成するか、別のプロジェクトファイルをテンプレートとして使用します（テンプレートの設定については、 **プロジェクトファイルのオプション** を参照してください）。
- **開く**... は、プロジェクトをファイル若しくは GeoPackage, PostgreSQL または Oracle データベースから開きます
- **閉じる** プロジェクトを閉じるか、最後に保存した状態に戻します。
- **保存** は、プロジェクトを `.qgs` または `.qgz` ファイル形式で、ファイル若しくは GeoPackage、PostgreSQL または Oracle データベースに保存します
- マップキャンバスを異なるファイル形式でエクスポートしたり、 **印刷レイアウト** を使用してより複雑な出力を行います。
- プロジェクトのプロパティやジオメトリ編集のスナップオプションを設定します。

表 7.1: プロジェクトメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
新規	Ctrl+N	プロジェクト	*QGIS* プロジェクトの紹介
テンプレートから新規作成			*QGIS* プロジェクトの紹介
開く...	Ctrl+O	プロジェクト	*QGIS* プロジェクトの紹介
開く			*QGIS* プロジェクトの紹介
GeoPackage...			*QGIS* プロジェクトの紹介
PostgreSQL...			*QGIS* プロジェクトの紹介
Oracle...			*QGIS* プロジェクトの紹介
最近使用したプロジェクト	Alt+J + R		*QGIS* プロジェクトの紹介
閉じる			*QGIS* プロジェクトの紹介
保存	Ctrl+S	プロジェクト	*QGIS* プロジェクトの紹介
名前を付けて保存...	Ctrl+Shift+S	プロジェクト	*QGIS* プロジェクトの紹介
保存			*QGIS* プロジェクトの紹介
テンプレート...			*QGIS* プロジェクトの紹介
GeoPackage...			*QGIS* プロジェクトの紹介
PostgreSQL...			*QGIS* プロジェクトの紹介
元に戻す...			
プロパティ...	Ctrl+Shift+P		プロジェクトのプロパティ
スナップオプション...			スナップとデジタイズのオプション
インポートとエクスポート			
地図を画像にエクスポート...			マップビューのエクスポート
地図を *PDF* にエクスポート...			マップビューのエクスポート
プロジェクトを *DXF* にエクスポート...			新しい *DXF* ファイルを作成する
DWG/DXF からレイヤをインポート...			*DXF* ファイルや *DWG* ファイルをインポートする
新規印刷レイアウト...	Ctrl+P	プロジェクト	地図のレイアウト
新規レポート...			レポートの作成
レイアウトマネージャ...		プロジェクト	地図のレイアウト
レイアウト			地図のレイアウト
モデル			モデルデザイナー
QGIS を終了	Ctrl+Q		

macOS では、 *QGIS* **を終了** コマンドは *QGIS*　 *QGIS* **を終了** （ Cmd+Q ） に対応します。

7.1.2 編集

編集 メニューには、レイヤの属性やジオメトリを編集するために必要なネイティブツールのほとんどが用意されています。menuselection:**編集** メニューのオプションを有効にするには、 編集モード切り替え をクリックして編集モードに切り替える必要があります（詳細は **編集** を参照してください）。

表 7.2: 編集メニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
元に戻す	Ctrl+Z	**デジタイジング**	**元に戻すとやり直し**
やり直す	Ctrl+Shift+Z	**デジタイジング**	**元に戻すとやり直し**
地物を切り取り	Ctrl+X	**デジタイジング**	**地物の切り取り、コピーと貼り付け**
地物をコピー	Ctrl+C	**デジタイジング**	**地物の切り取り、コピーと貼り付け**
地物を貼り付け	Ctrl+V	**デジタイジング**	**地物の切り取り、コピーと貼り付け**
新規レイヤに地物を貼り付け			**属性テーブルの操作**
新規ベクタレイヤ...			**属性テーブルの操作**
一時スクラッチレイヤ...	Ctrl+Alt+V		**属性テーブルの操作**
選択物を削除		**デジタイジング**	**選択地物の削除**
選択			**地物の選択**
地物を選択		**選択**	**地物の選択**
ポリゴンによる地物選択		**選択**	**地物の選択**
フリーハンドによる地物選択		**選択**	**地物の選択**
円による地物選択		**選択**	**地物の選択**
値による地物選択...	F3	**選択**	**地物の選択**
式による地物選択...	Ctrl+F3	**選択**	**地物の選択**
全レイヤの選択を解除	Ctrl+Alt+A	**選択**	**地物の選択**
アクティブレイヤの選択を解除	Ctrl+Shift+A	**選択**	**地物の選択**
地物を再選択			**地物の選択**
すべての地物を選択	Ctrl+A	**選択**	**地物の選択**
地物選択を反転		**選択**	**地物の選択**

次のページに続く

表 7.2 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
レコードを追加	Ctrl+.	デジタイジング	
点地物を追加する	Ctrl+.	デジタイジング	地物の追加
線の地物を追加	Ctrl+.	デジタイジング	地物の追加
ポリゴン地物を追加	Ctrl+.	デジタイジング	地物の追加
円形ストリングを追加		シェープデジタイジング	*Circular string by radius*
半径指定による円形ストリングの追加		シェープデジタイジング	*Circular string by radius*
円を追加		シェープデジタイジング	円を描く
2 点から円		シェープデジタイジング	円を描く
3 点から円		シェープデジタイジング	円を描く
3 本の接線から円		シェープデジタイジング	円を描く
2 本の接線と点から円		シェープデジタイジング	円を描く
中心点と別の点で円		シェープデジタイジング	円を描く
長方形を追加		シェープデジタイジング	長方形を描く
領域範囲から長方形		シェープデジタイジング	長方形を描く

次のページに続く

表 7.2 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
中心と点から長方形		シェープデジタイジング	長方形を描く
3 点からの長方形 （距離）		シェープデジタイジング	長方形を描く
3 点からの長方形 （投影）		シェープデジタイジング	長方形を描く
正多角形を追加		シェープデジタイジング	正多角形を描く
中心と点から正多角形		シェープデジタイジング	正多角形を描く
中心と角から正多角形		シェープデジタイジング	正多角形を描く
2 点から正多角形		シェープデジタイジング	正多角形を描く
楕円を追加		シェープデジタイジング	楕円を描く
中心と *2* 点から楕円		シェープデジタイジング	楕円を描く
中心と点から楕円		シェープデジタイジング	楕円を描く
領域範囲から楕円		シェープデジタイジング	楕円を描く
フォーカスから楕円		シェープデジタイジング	楕円を描く
注記を追加			注記ツール
注記		注記	注記ツール
注記フォーム		注記	注記ツール

次のページに続く

表 7.2 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
HTML 注記		注記	注記ツール
SVG 注記		注記	注記ツール
属性を編集			
選択地物の属性変更		デジタイジング	属性値の編集
選択地物の属性結合		高度なデジタイズ	選択地物の属性結合
ジオメトリを編集			
地物を移動		高度なデジタイズ	地物の移動
地物をコピー/移動		高度なデジタイズ	地物の移動
地物を回転		高度なデジタイズ	地物を回転
地物をスケール		高度なデジタイズ	地物をスケーリング
地物を簡素化		高度なデジタイズ	地物を簡素化
リングを追加		高度なデジタイズ	リングを追加
部分を追加		高度なデジタイズ	部分を追加
リングを充填		高度なデジタイズ	リングを充填
リングを削除		高度なデジタイズ	リングを削除
パートを削除		高度なデジタイズ	部分を削除
地物を変形		高度なデジタイズ	地物の変形
曲線をオフセット		高度なデジタイズ	曲線のオフセット
地物を分割		高度なデジタイズ	地物を分割
パートを分割		高度なデジタイズ	部分の分割

次のページに続く

表 7.2 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
選択地物を結合		**高度なデジタイズ**	**選択地物の結合**
頂点ツール（全レイヤ）		**デジタイジング**	**頂点ツール**
頂点ツール（現在のレイヤ）		**デジタイジング**	**頂点ツール**
線の向きを反転		**高度なデジタイズ**	**線の向きの反転**
地物をトリム/延長		**高度なデジタイズ**	**地物をトリム/延長**
点のシンボルを回転		**高度なデジタイズ**	**点のシンボルを回転**
点のシンボルをオフセット		**高度なデジタイズ**	**点のシンボルのオフセット**

選択されたレイヤのジオメトリタイプ（ポイント、ポリライン、ポリゴンなど）に依存するツールは、タイプに応じて有効化されます。

表 7.3: 「地物を移動」のジオメトリに応じたアイコン

メニューオプション	ポイント	ポリライン	ポリゴン
地物を移動			
地物をコピー/移動			

7.1.3 ビュー

地図はマップビューにレンダリングされます。これらのビューは、**ビュー** ツールを使用して操作することができます。例えば、次のことができます:

- メインマップキャンバスの隣に新しい 2D または 3D マップビューを作成する
- 任意の場所に **ズームまたはパン** する
- 表示された地物の属性やジオメトリをクエリする
- プレビューモードや注釈、装飾を使用して、マップビューを充実させる
- パネルまたはツールバーにアクセスする

このメニューでは、次のようなアクションを使用して QGIS のインターフェイス自体を再編成することもできます。

- **フルスクリーン切り替え**: タイトルバーを非表示にして全画面表示にします

- **パネル表示切り替え**: 有効な **パネル** の表示/非表示を切り替えます。地物をデジタイズするとき（キャンバスを最大限に表示するため）や、QGIS のメインキャンバスを使ったプレゼンテーション（投影/録画）に便利です
- **地図のみ切り替え**: パネル、ツールバー、メニュー、ステータスバーを非表示にし、マップキャンバスのみを表示します。全画面オプションと組み合わせると、画面に地図のみが表示されます

表 7.4: ビューメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
 **新規マップビュー**	Ctrl+M	**ナビゲーション**	*2D* **マップビュー**
3D **マップビュー**			*3D* **マップビュー**
 **新規** *3D* **マップビュー**	Ctrl+Alt+M	**ナビゲーション**	*3D* **マップビュー**
3D **マップビューを管理**			*3D* **マップビュー**
 **地図を移動**		**ナビゲーション**	**マップビューについて詳しくみる**
 **選択部分にパン**		**ナビゲーション**	**マップビューについて詳しくみる**
 **拡大**	Ctrl+Alt++	**ナビゲーション**	**マップビューについて詳しくみる**
 **縮小**	Ctrl+Alt+-	**ナビゲーション**	**マップビューについて詳しくみる**
 **地物情報を表示**	Ctrl+Shift+I	**属性**	**地物の識別**
計測		**属性**	**計測**
 **線の長さを測る**	Ctrl+Shift+M	**属性**	**計測**
 **面積を測る**	Ctrl+Shift+J	**属性**	**計測**
 **角度を測る**		**属性**	**計測**
 **角度を測る**		**属性**	**計測**
 **統計サマリー**		**属性**	**統計量の出力パネル**
 **標高断面図**			**標高断面図ビュー**
 **全域表示**	Ctrl+Shift+F	**ナビゲーション**	**マップビューについて詳しくみる**
 **選択部分にズーム**	Ctrl+J	**ナビゲーション**	**マップビューについて詳しくみる**
 **レイヤの領域にズーム**		**ナビゲーション**	**マップビューについて詳しくみる**
 **ネイティブ解像度にズーム（***100%***）**		**ナビゲーション**	**マップビューについて詳しくみる**

次のページに続く

表 7.4 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
前の領域へズーム		ナビゲーション	マップビューについて詳しくみる
次の表示領域にズーム		ナビゲーション	マップビューについて詳しくみる
地図整飾	Alt+V + D		地図の整飾
グリッド...			グリッド
スケールバー...			スケールバー
画像...			画像
方位記号...			方位記号
タイトルラベル...			タイトルラベル
著作権ラベル...			著作権ラベル
レイアウト範囲...			レイアウト範囲
プレビューモード			
通常（*Normal*）			
モノクロシミュレート（*Monochrome*）			
全色盲シミュレート（*Achromatopsia*）			
赤色盲シミュレート（*Protanope*）			
緑色盲シミュレート（*Deuteranope*）			
黄青色盲シミュレート（*Tritanopia*）			
地図の *Tips* を表示		属性	表示名プロパティ
新規空間ブックマーク...	Ctrl+B	ナビゲーション	地図上の範囲のブックマーク
空間ブックマークを表示	Ctrl+Shift+B	ナビゲーション	地図上の範囲のブックマーク
空間ブックマーク・マネージャを表示			地図上の範囲のブックマーク
再読み込み	F5	ナビゲーション	
レイヤの可視性			レイヤパネル
すべてのレイヤを表示	Ctrl+Shift+U		レイヤパネル
すべてのレイヤを隠す	Ctrl+Shift+H		レイヤパネル
選択レイヤを表示			レイヤパネル
選択レイヤを隠す			レイヤパネル
選択レイヤを切り替え			レイヤパネル
選択レイヤを個別切り替え			レイヤパネル
選択されてないレイヤを隠す			レイヤパネル

次のページに続く

表 7.4 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
パネル			パネルとツールバー
ツールバー			パネルとツールバー
フルスクリーン切り替え	F11		
パネル表示切り替え	Ctrl+Tab		
地図のみ切り替え	Ctrl+Shift+Tab		

Linux KDE では、**パネル** 、 **ツールバー** および **フルスクリーン切り替え** は、 **設定** メニュー内にあります。

7.1.4 レイヤ

レイヤ メニューには、新しいデータソースを **作成** したり、プロジェクトに **追加** したり、**変更を保存** したりするための多数のツールが用意されています。同じデータソースを使って、以下のこともできます:

- **レイヤを複製** レイヤを複製してコピーを作成し、名前やスタイル（シンボロジ、ラベル、...)、テーブル結合などを変更することができます。コピーはオリジナルと同じデータソースを使います。
- **コピー** と **貼り付け** あるプロジェクトのレイヤまたはグループを別のところへ新たなインスタンスとして貼り付けます。プロパティは元のものとは独立して変更することができます。 **複製** に関しては、レイヤは同じデータソースに基づいたままです。
- **レイヤとグループを埋め込む**... 他のプロジェクトから、変更することができない読み取り専用のコピーとしてレイヤあるいはグループを読み込みます（ **外部プロジェクトからのレイヤの埋め込み** を参照してください）

レイヤ メニューには、レイヤのプロパティ（スタイル、スケール、CRS...）を設定したり、コピー、貼り付けするツールも含まれています。

表 7.5: レイヤメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
データソースマネージャ	Ctrl+L	**データソースマネージャ**	**データを開く**
レイヤを作成			**新しいベクタレイヤを作成する**
新規 *GeoPackage* **レイヤ**...	Ctrl+Shift+N	**データソースマネージャ**	**新しい** *GeoPackage* **レイヤを作成する**
新規シェープファイルレイヤ...		**データソースマネージャ**	**新しいシェープファイルレイヤを作成する**
新規 *SpatiaLite* **レイヤ**...		**データソースマネージャ**	**新しい** *SpatiaLite* **レイヤを作成する**

次のページに続く

表 7.5 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
新規一時スクラッチレイヤ...		データソースマネージャ	新しい一時スクラッチレイヤを作成する
新規メッシュレイヤ...		データソースマネージャ	新しいメッシュレイヤを作成する
新規 *GPX* レイヤ...		データソースマネージャ	新しい *GPX* レイヤを作成する
新規仮想レイヤ...		データソースマネージャ	仮想レイヤを作成する
レイヤを追加			データを開く
ベクタレイヤを追加...	Ctrl+Shift+V	レイヤ管理	ファイルからレイヤを読み込む
ラスタレイヤを追加...	Ctrl+Shift+R	レイヤ管理	ファイルからレイヤを読み込む
メッシュレイヤを追加...		レイヤ管理	メッシュレイヤを読み込む
CSV テキストレイヤを追加...	Ctrl+Shift+T	レイヤ管理	区切りテキストファイルをインポートする
PostGIS レイヤを追加...	Ctrl+Shift+D	レイヤ管理	データベース関連ツール
SpatiaLite レイヤを追加...	Ctrl+Shift+L	レイヤ管理	*SpatiaLite* レイヤ
MS SQL サーバーレイヤを追加...		レイヤ管理	データベース関連ツール
Oracle Spatial レイヤを追加...		レイヤ管理	データベース関連ツール
SAP HANA 空間レイヤを追加...		レイヤ管理	データベース関連ツール
仮想レイヤを追加/編集...		レイヤ管理	仮想レイヤを作成する
WMS/WMTS を追加...	Ctrl+Shift+W	レイヤ管理	*WMS/WMTS* レイヤを読み込む
XYZ レイヤを追加...			*XYZ* タイルサービスを利用する
WCS レイヤを追加...		レイヤ管理	*WCS* クライアント
WFS レイヤを追加...		レイヤ管理	*WFS* および *WFS-T* クライアント
ArcGIS REST Server レイヤを追加...		レイヤ管理	*ArcGIS REST Servers* を利用する
ベクタタイルレイヤを追加...			ベクタタイルサービスを利用する
点群レイヤを追加...			点群の操作
GPX レイヤを追加...			*GNSS/GPS* データの導入

次のページに続く

表 7.5 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
レイヤとグループを埋め込む...			外部プロジェクトからのレイヤの埋め込み
レイヤ定義ファイルから追加...			レイヤ定義ファイル
ジオリファレンサ...			ジオリファレンサ
スタイルをコピー			レイヤのプロパティの保存および共有
スタイルを貼り付け			レイヤのプロパティの保存および共有
レイヤをコピー			
レイヤ/グループを貼り付け			
属性テーブルを開く	F6	属性	属性テーブルの操作
属性テーブルをフィルタ			属性テーブルの操作
属性テーブルを開く（選択地物）	Shift+F6	属性	属性テーブルの操作
属性テーブルを開く（可視地物）	Ctrl+F6	属性	属性テーブルの操作
属性テーブルを開く（編集済み地物）		属性	属性テーブルの操作
編集モード切り替え		デジタイジング	既存レイヤのデジタイズ
レイヤ編集内容を保存		デジタイジング	レイヤ編集内容の保存
現在の編集		デジタイジング	レイヤ編集内容の保存
選択レイヤを保存		デジタイジング	レイヤ編集内容の保存
選択レイヤをロールバック		デジタイジング	レイヤ編集内容の保存
選択レイヤの編集キャンセル		デジタイジング	レイヤ編集内容の保存
全レイヤを保存		デジタイジング	レイヤ編集内容の保存
全レイヤをロールバック		デジタイジング	レイヤ編集内容の保存
全レイヤ編集キャンセル		デジタイジング	レイヤ編集内容の保存
名前を付けて保存...			既存のレイヤから新しいレイヤを作成する
レイヤ定義ファイルとして保存...			レイヤ定義ファイル
レイヤ/グループを削除	Ctrl+D		
レイヤを複製			
レイヤを表示するスケールを設定			表示縮尺セレクタ
レイヤの *CRS* を設定	Ctrl+Shift+C		レイヤの座標参照系
レイヤの *CRS* をプロジェクトに設定			プロジェクトの座標参照系

次のページに続く

表 7.5 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
レイヤのプロパティ...			**ベクタプロパティダイアログ**, **ラスタプロパティダイアログ**, **メッシュデータセットのプロパティ**, **点群のプロパティ**, **ベクタタイルデータセットのプロパティ**
フィルタ...	Ctrl+F		**クエリビルダ**
ラベル			**ラベルプロパティ**
全体図に表示			**全体図パネル**
全体図にすべて表示			**全体図パネル**
全体図からすべて隠す			**全体図パネル**

7.1.5 設定

表 7.6: 設定メニューの項目

メニューオプション	リファレンス
ユーザープロファイル	**ユーザープロファイルの操作**
default	**ユーザープロファイルの操作**
アクティブなプロファイルフォルダを開く	**ユーザープロファイルの操作**
新規プロファイル...	**ユーザープロファイルの操作**
スタイルマネージャ...	**スタイルマネージャ**
カスタム投影法...	**カスタム座標参照系**
キーボードショートカット...	**キーボードショートカット**
インタフェースのカスタマイズ...	**インタフェースのカスタマイズ**
オプション...	**オプション**

Linux KDE では、**設定** メニュー内に **パネル** や **ツールバー** 、**フルスクリーン切り替え** といったツールがあります。

7.1.6 プラグイン

表 7.7: プラグインメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
プラグインの管理とインストール...			**プラグインダイアログ**
Python **コンソール**	Ctrl+Alt+P	**プラグイン**	*QGIS Python* **コンソール**

初めて QGIS を起動したときには、すべてのコアプラグインがロードされるわけではありません。

7.1.7 ベクタ

以下はすべてのコアプラグインが有効になっている場合の **ベクタ** メニューです。

表 7.8: ベクタメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
ジオメトリをチェック...			**ジオメトリチェッカープラグイン**
トポロジチェッカー		**ベクタ**	**トポロジチェッカープラグイン**
空間演算ツール	Alt+O + G		
バッファ *(buffer)...*			**バッファ（***buffer***）**
切り抜く *(clip)...*			**切り抜く**
凸包 *(convex hull)...*			**凸包（***convex hull***）**
差分 *(difference)...*			**差分**
融合 *(dissolve)...*			**融合（***dissolve***）**
交差 *(Intersect)...*			**交差**
対称差 *(symmetrical difference)...*			**対称差**
和集合 *(union)...*			**和集合**
選択地物の周辺ポリゴンを融合 *(eliminate)...*			**選択物の隣接ポリゴンの融合（***eliminate***）**
ジオメトリツール	Alt+O + E		
重心...			**重心**
シングルパートをマルチパートに集約...			**シングルパートをマルチパートに集約**
頂点を抽出...			**頂点を抽出**
マルチパートをシングルパートに変換...			**マルチパートをシングルパートに変換**

次のページに続く

表 7.8 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
ポリゴンを線に変換...			ポリゴンを線に変換
ジオメトリを簡素化...			ジオメトリの簡素化
有効性チェック...			有効性チェック
ドロネー三角分割...			ドロネー三角分割
頂点の高密度化（個数ベース）...			頂点の高密度化（個数ベース）
ジオメトリ属性を追加...			ジオメトリ属性を追加
線をポリゴンに変換（*lines to polygons*）...			線をポリゴンに変換（*lines to polygons*）
ボロノイ多角形...			ボロノイ多角形
解析ツール	Alt+O + A		
線の交差（*intersect*）...			線の交差
加重平均座標（重心の平均）...			加重平均座標（重心の平均）
属性の基本統計量（*HTML* 出力）...			属性の基本統計量
ポリゴン内の点の数...			ポリゴン内の点の数
距離行列（*distance matrix*）...			距離行列（*distance matrix*）
ユニーク値のリスト...			ユニーク値のリスト
最近傍解析...			最近傍解析
線長の合計...			線長の合計
データ管理ツール	Alt+O + D		
ベクタレイヤをマージ...			ベクタレイヤのマージ
ベクタレイヤを再投影...			レイヤの再投影
空間インデックスを作成...			空間インデックスを作成
属性の空間結合...			属性の空間結合
属性でレイヤを分割...			属性でレイヤを分割
調査ツール	Alt+O + R		
グリッドを作成...			グリッドを作成
レイヤ範囲を抽出...			レイヤ範囲を抽出
指定範囲にランダム点群...			ランダム点群
ポリゴン内部にランダム点群（*in*）...			ポリゴン内部にランダム点群
線の上のランダム点群...			線に沿ったランダム点群
場所による選択...			場所による選択
レイヤ領域にランダム点群...			入力レイヤの領域にランダム点群

次のページに続く

表 7.8 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
ポリゴン内部にランダム点群（*inside*）...			**ポリゴン内部にランダム点群**
ランダム選択...			**ランダム選択**
地物のグループ別ランダム選択...			**地物のグループ別ランダム選択**
規則的点群（*regular points*）...			**規則的な点群**

デフォルトでは、QGIS は **プロセシング** アルゴリズムをサブメニューでグループ化された **ベクタ** メニューに追加します。これは、ベクタレイヤに対する様々なプロバイダからの多数の一般的な GIS タスクへのショートカットを提供します。これらのサブメニューが全て使用できない場合には、 **プラグイン** **プラグインの管理とインストール...** で Processing プラグインを有効化してみてください。

アルゴリズムのリストとそのメニューは、任意のプロセシングアルゴリズム（ **プロセシングフレームワークを設定する** 参照）や、外部の **プラグイン** を用いて変更や拡張が可能です。

7.1.8 ラスタ

以下はすべてのコアプラグインが有効になっている場合の **ラスタ** メニューです。

表 7.9: ラスタメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
ラスタ計算機...			**ラスタ計算機**
ラスタを揃える...			**ラスタを揃える**
解析			
傾斜方位（*aspect*）...			**傾斜方位**
***nodata* 値を内挿値で埋める...**			***nodata* を埋める**
グリッド（移動平均）...			**グリッド（移動平均）**
グリッド（データメトリクス）...			**グリッド（データメトリクス）**
グリッド（累乗逆距離加重法）...			**グリッド（累乗逆距離加重法）**
グリッド（最近傍法）...			**グリッド（最近傍探索 *IDW*）**
純黒化（*near black*）...			**純黒化** *(near black)*
陰影図（*hillshade*）...			**陰影図（*hillshade*）**
特定値までの距離（*proximity*）...			**特定値までの距離** *(raster distance)*
粗度（*roughness*）...			**粗度** *(roughness)*

次のページに続く

表 7.9 – 前のページからの続き

メニューオプション	ショートカット	ツールバー	リファレンス
ふるい（*sieve*）...			ふるい
傾斜（*slope*）...			傾斜
TPI（*Topographic Position Index*）...			地形位置指数 *(TPI)*
TRI（*Terrain Ruggedness Index*）...			*Terrain Ruggedness Index (TRI)*
投影法			
投影法を設定...			投影法を設定
投影法を抽出...			投影法を抽出
再投影（*warp*）...			再投影 *(Warp)*
その他			
仮想ラスタを構築...			仮想ラスタを構築
ラスタの情報...			ラスタの情報
結合（*gdal_merge*）...			結合
全体図を作成（ピラミッド）...			全体図を作成（ピラミッド）
タイルインデックス...			タイルインデックス
抽出			
範囲を指定して切り抜く...			範囲を指定して切り抜く
マスクレイヤで切り抜く...			マスクレイヤで切り抜く
等高線（*contour*）...			等高線
変換			
PCT を *RGB* に変換...			*PCT* を *RGB* に変換
ラスタをベクタ化（*polygonize*）...			ラスタをベクタ化 *(polygonize)*
ベクタをラスタ化（*rasterize*）...			ベクタをラスタ化
RGB を *PCT* に変換...			*RGB* を *PCT* に変換
形式変換（*gdal_translate*）...			形式変換 *(convert format)*

デフォルトでは、QGIS は **プロセシング** アルゴリズムをサブメニューでグループ化された **ラスタ** メニューに追加します。これは、ラスタレイヤに対する様々なプロバイダからの多数の一般的な GIS タスクへのショートカットを提供します。これらのサブメニューが全て使用できない場合には、 **プラグイン** **プラグインの管理とインストール**... で Processing プラグインを有効化してみてください。

アルゴリズムのリストとそのメニューは、任意のプロセシングアルゴリズム（ **プロセシングフレームワークを設定する** 参照）や、外部の **プラグイン** を用いて変更や拡張が可能です。

7.1.9 データベース

以下はすべてのコアプラグインが有効になっている場合の **データベース** メニューです。データベースに関連したプラグインが有効化されていない場合には、**データベース** メニューには何もありません。

表 7.10: データベースメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
オフライン編集...	Alt+D + O		**プラグインをオフラインで編集する**
 **オフラインプロジェクトに変換...**		**データベース**	**プラグインをオフラインで編集する**
 **同期**		**データベース**	**プラグインをオフラインで編集する**
 *DB* **マネージャ...**		**データベース**	*DB* **マネージャプラグイン**

初めて QGIS を起動したときには、すべてのコアプラグインがロードされるわけではありません。

7.1.10 Web

以下はすべてのコアプラグインが有効になっている場合の *Web* メニューです。Web に関連したプラグインが有効化されていない場合には、*Web* メニューには何もありません。

表 7.11: Web メニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
MetaSearch	Alt+W + M		*MetaSearch Catalog Client*
 *Metasearch*		*Web*	*MetaSearch Catalog Client*
Help			*MetaSearch Catalog Client*

初めて QGIS を起動したときには、すべてのコアプラグインがロードされるわけではありません。

7.1.11 メッシュ

メッシュ メニューは、 **メッシュレイヤ** を操作するために必要なツールを提供します。サードパーティプラグインはこのメニューに項目を追加することができます。

表 7.12: メッシュメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
メッシュ計算機...			メッシュ計算機
面と点を再インデックス化			メッシュの再インデックス化

7.1.12 プロセシング

表 7.13: プロセシングメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
ツールボックス	Ctrl+Alt+T		ツールボックス
グラフィカルモデラー...	Ctrl+Alt+G		モデルデザイナー
履歴...	Ctrl+Alt+H		履歴マネージャ
結果ビューア	Ctrl+Alt+R		外部アプリケーションの設定
In-Place 編集			プロセシングによるレイヤのインプレース修正

初めて QGIS を起動したときには、すべてのコアプラグインがロードされるわけではありません。

7.1.13 ヘルプ

表 7.14: ヘルプメニューの項目

メニューオプション	ショートカット	ツールバー	リファレンス
QGIS ヘルプ	F1	ヘルプ	
API ドキュメント			
プラグイン			
問題を報告			
有償サポートが必要ですか？			
QGIS のホームページ	Ctrl+H		
QGIS のバージョン情報			
QGIS について			
QGIS 維持会員			

7.1.14 QGIS

このメニューは、 Mac OS でのみ使用可能で、OS に関連するコマンドが含まれています。

表 7.15: QGIS メニュー項目

メニューオプション	ショートカット
Preferences	
QGIS **について**	
QGIS **を隠す**	
すべてを表示	
他を非表示	
QGIS **を終了**	Cmd+Q

他のプラットフォームでは *Preferences* は **設定** **オプション** 、 *About QGIS* は **ヘルプ** *QGIS* **について** 、 *Quit QGIS* は **プロジェクト** *QGIS* **を終了** に対応します。

7.2 パネルとツールバー

ビュー メニュー（ は **設定**）で QGIS のパネル（**パネル** ）やツールバー（**ツールバー** ）の オン・オフを切り替えられます。これらのいずれかを有効化（あるいは無効化）するには、メニューバーやツールバーを右クリックし、必要な項目を選択します。パネルやツールバーは、QGIS インターフェース の好きな場所に移動して配置できます。このリストは、**コアプラグインまたは外部プラグイン** の有効化で拡張することもできます。

7.2.1 ツールバー

ツールバーはメニュー内の機能の大半にアクセスできるだけでなく、さらにマップを操作するためのツールにもアクセスできます。ツールバーの各アイテムにはポップアップヘルプがあります。アイテムの上にマウスを置くと、ツールの目的に関する簡単な説明が表示されます。

利用可能なツールバーは以下のとおりです：

表 7.16: QGIS のツールバー

ツールバーの名称	ツールの主な参照先
高度なデジタイズ	**高度なデジタイズ**
注記	**注記ツール**
属性	**属性テーブルの操作**, **一般ツール**
データソースマネージャ	**データソースの管理**
データベース	*DB* **マネージャプラグイン**
デジタイジング	**既存レイヤのデジタイズ**
GRASS	*GRASS GIS* **の統合**
ヘルプ	
ラベル	**ラベルツールバー**
レイヤ管理	**データを開く**
ナビゲーション	**マップビューについて詳しくみる**
メッシュデジタイジング	**メッシュレイヤを編集する**
プラグイン	**プラグイン**
プロジェクト	**プロジェクトファイルでの作業**, **地図のレイアウト**, **スタイルライブラリ**
プロセシングツールボックスパネル	**プロセシングフレームワークを設定する**
ラスタ	**プラグイン**
選択	**地物の選択**
シェープデジタイジングツールバー	**シェープデジタイジング**
スナップツールバー	**スナップ許容範囲と検索半径の設定**
ベクタ	**プラグイン**
Web	**プラグイン**, *MetaSearch Catalog Client*

注釈: サードパーティーのプラグインは、デフォルトのツールバーを独自のツールで拡張したり、独自のツールバーを提供することがあります。

Tip: ツールバーの復元

誤ってツールバーを非表示にしてしまった場合には、**ビュー　ツールバー**　（  の場合は **設定　ツールバー**　）を使ってもとに戻すことができます。何らかの要因でツールバー（またはその他のパネル）が完全に消えてしまった場合には、 *GUI* **の初期状態の復元** に元に戻すためのヒントがあります。

7.2.2 パネル

QGIS にはさまざまなパネルがあります。パネルとは、より複雑な作業（オプションの選択、ボックスのチェック、値の入力など）を実行するための特別なウィジェットです。

以下は QGIS が提供するデフォルトのパネルの一覧です。

表 7.17: QGIS のパネル

パネルの名称	ショートカット	リファレンス
高度なデジタイズ	Ctrl+4	**高度なデジタイズパネル**
ブラウザ	Ctrl+2	**ブラウザパネル**
ブラウザ *(2)*		**ブラウザパネル**
デバッグ開発ツール	F12	**デバッグ開発ツールパネル**
標高断面図		
ジオメトリ検証		**デジタイズプロパティ**
GPS **情報**	Ctrl+0	**ライブ** *GPS* **追跡**
GRASS **ツール**		*GRASS GIS* **の統合**
レイヤ順序	Ctrl+9	**レイヤ順序パネル**
レイヤスタイル	Ctrl+3	**レイヤスタイル設定パネル**
レイヤ	Ctrl+1	**レイヤパネル**
ログメッセージ		**ログメッセージパネル**
全体図	Ctrl+8	**全体図パネル**
プロセシングツールボックス		**ツールボックス**
結果ビューア		**ツールボックス**
スナップとデジタイジングオプション		**スナップ許容範囲と検索半径の設定**
空間ブックマークマネージャ	Ctrl+7	**地図上の範囲のブックマーク**
統計量の出力	Ctrl+6	**統計量の出力パネル**
時系列コントローラ		**時系列コントローラパネル**
タイルスケール		**タイルセット**
元に戻す/やり直す	Ctrl+5	**元に戻す/やり直すパネル**
頂点エディタ		**頂点エディタパネル**

7.3 ステータスバー

ステータスバーにはマップビューと処理済みアクションまたは使用可能なアクションに関する一般的な情報が表示され、マップビューを管理するためのツールが提供されています。

7.3.1 ロケータバー

ステータスバーの左には、クイック検索ウィジェットであるロケータバーがあり、QGIS の任意の機能やオプションを検索し実行できます：

1. ロケータ検索バーをアクティブにするには、テキストウィジェット内をクリックするか、 `Ctrl+K` を押します。
2. 探しているアイテムに関連したテキストを入力します（名前、タグ、キーワード等）。デフォルトでは、有効になっているロケータフィルタに対して検索結果が返されますが、 **ロケータフィルタ** の接頭辞をテキストの前に付けることで、検索を特定のスコープに限定することもできます。例えば、 `l cad` と入力すると、名前に `cad` を含むレイヤのみが返されます。

   このフィルタは、ロケータウィジェットにアクセスした際にメニュー内に表示されるものをダブルクリックすることでも選択できます。
3. 検索結果をクリックすると、アイテムのタイプに応じて対応するアクションを実行します。

Tip: 検索をアクティブレイヤの特定のフィールドに限定する

デフォルトでは、「アクティブレイヤの地物」フィルタ（ `f` ）による検索は、レイヤの属性テーブル全体にわたって行われます。この検索は、 `@` 接頭辞を使用することで特定のフィールドに限定することができます。例えば、 `f @name sal` あるいは `@name sal` は、 "name" 属性に 'sal' という文字列を持つ地物のみを返します。テキストを書く際の自動補完が有効となり、候補は `Tab` キーを使用して適用することができます。

検索フィールドに関するより高度な制御は、レイヤの **属性** タブ内で行うことができます。詳細については **属性プロパティ** を参照してください。

検索はスレッドを使用して処理されるため、たとえ低速の検索フィルタがインストールされていても、常に可能な限り早く結果が得られるようになっています。また、フィルタによって検索結果が見つかると、すぐに表示されるようになっています。例えば、ファイル検索フィルタがファイルツリーをスキャンすると、結果が一つ見つかるごとに表示されます。これにより、非常に遅い検索フィルタ（オンラインサービスを使用するものなど）が存在する場合でも、UI は常に応答できるようになっています。

注釈: Nominatim ロケータツールは、OpenStreetMap Nominatim の usage policy に関連して、上記とは異なる動作をします（自動補完検索ができない、結果の取得が遅れる等）。

Tip: ロケータの設定へのクイックアクセス

ステータスバーのロケータウィジェット内の  アイコンをクリックし、使用可能なフィルタのリストを表示したら、 **設定...** エントリをクリックすると、 **設定** **オプション...** メニューの **ロケータ** タブが開きます。

7.3.2 アクションの報告

ロケータバーの隣には、必要に応じて実行した操作の概要（レイヤ内の地物を選択する、レイヤを削除する、パンの距離と方向）や、マウスカーソルを上に載せているツールの長い説明（すべてのツールで利用できるわけではない）が表示されます。

ラスタレイヤの統計情報の収集、プロセシングアルゴリズムの実行、マップビューでの複数レイヤのレンダリングなど時間がかかるの操作の場合には、ステータスバーにプログレスバーが表示されます。

7.3.3 マップキャンバスのコントロール

 **座標** オプションは、マップビューをマウスが移動している間、マウスの現在位置を追跡して表示します。座標の単位（および精度）は、 **プロジェクト** **プロパティ...** **一般情報** タブで設定できます。テキストボックスの左にある小さなボタンをクリックすると、座標オプションと  **範囲** オプションが切り替わります。範囲オプションでは、現在のマップビューの左下隅と右上隅座標がマップの単位で表示されます。

座標表示の隣には **縮尺** 表示があります。これはマップビューの縮尺を表します。縮尺セレクタがあり、**定義済み縮尺とカスタム縮尺** を選択できます。

縮尺表示の右側で、  ボタンを押して縮尺をロックすると、拡大レベルを使って拡大・縮小することができます。拡大レベルを使うと、地図の縮尺を変えずに地図を拡大できるので、ラベルや記号の位置を正確に微調整しやすくなります。拡大率はパーセントで表示されます。もし **拡大レベル** の値が 100%であれば、現在の地図は拡大されず、モニターの解像度（DPI）に対して正確な縮尺で描画されます。デフォルトの拡大率は **設定** **オプション** **レンダリング** **レンダリング動作** で定義することができ、高解像度のスクリーンで小さなシンボルを拡大する際にとても便利です。さらに、 **設定** **オプション** **キャンバスと凡例** *DPI* の設定は、QGIS が各モニタの物理 DPI を尊重するか、システム全体の論理 DPI を使用するかを制御します。

拡大ツールの右側では、地図ビューの現在の時計回りの回転を度で定義できます。

ステータスバーの右側にある  **レンダ** チェックボックスを使うと、マップビューのレンダリングを一時的に停止することができます（セクション **地図のレンダリングの制御** を参照）。

 **レンダ** 機能の右側には、現在のプロジェクト CRS を示す  *EPSG:code* ボタンがあります。これをクリックすると **プロジェクトのプロパティ** ダイアログが開き、マップビューの再投影やその他のプロジェクトのプロパティを調整することができます。

Tip: 地図キャンバスにおける正しい縮尺を計算する

QGIS を起動した際のデフォルトの CRS は `WGS 84 (EPSG 4326)` で、単位は度です。これは、QGIS はレイヤ内のあらゆる座標が度単位で指定されているものと解釈することを意味します。正しい縮尺値を得る

には、**プロジェクト　プロパティ**... の **一般** タブで単位の設定を手動で変更する（例：メートル）か、上で述べた  EPSG:code アイコンを使います。後者の場合、単位はプロジェクトの投影法で指定されたものに設定されます（例： `+units=us-ft` ）

なお、起動時の CRS の選択は **設定**　**オプション**　*CRS* **の扱い** で設定することができます。

7.3.4 メッセージ

その横にある  メッセージ ボタンをクリックすると、**ログメッセージパネル** が開きます。ここには基本的なプロセス（QGIS の起動、プラグインのロード、プロセシングツールなど）に関する情報があります。

第8章　ブラウザパネル

QGIS ブラウザパネルは、QGIS リソースの閲覧や検索、 中身の確認、コピーや読み込みに最適なツールです。QGIS が処理方法を認識しているリソースのみがブラウザパネルに表示されます。

ブラウザパネル で説明されているように、ブラウザパネルを使用するとデータの検索や中身の確認、データの追加を行うことができます。さらに、ブラウザパネルはプロジェクトファイル、Python スクリプトやプロセシングスクリプト、プロセシングモデルなど、多くの QGIS リソースのドラッグ＆ドロップをサポートしています。

Python スクリプトやプロセシングスクリプト、プロセシングモデルは、編集するために外部エディタやグラフィカルモデラーで開くこともできます。

レイヤ パネルからレイヤを **ブラウザ** パネルの例えば GeoPackage や PostGIS データベースへドラッグ＆ドロップすることができます。

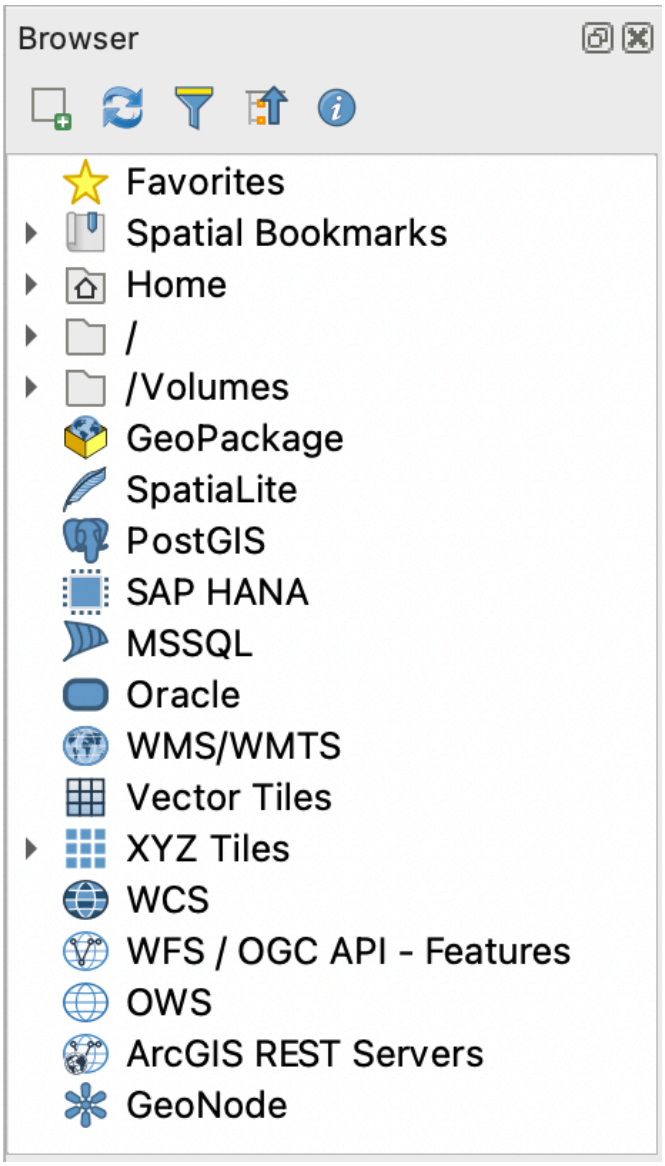


図 8.1: ブラウザパネル

ブラウザパネル (図 8.1) は、いくつかの固定トップレベルエントリを持った展開可能な階層として系統的に整理されています。階層にはブラウザで扱われるリソースがまとめられています。ノードエントリは、エントリ名の左側にある をクリックすることで展開されます。ブランチは をクリックすることで折りたたまれます。 全て折りたたむ ボタンを押すと、すべてのトップレベルエントリが折りたたまれます。

設定　インタフェースのカスタマイズ では機能を無効化することができます。例えば、ブラウザに Python スクリプトを表示したくない場合には *Browser　py* エントリのチェックを外し、ブラウザ内のホームフォルダを消したい場合には *Browser　special:Home* エントリのチェックを外します。

フィルタ（ ブラウザのフィルタ ）を使用して、エントリ名（階層内のリーフエントリとノードエントリの両方）に基づいて検索できます。フィルタテキストフィールドの横にある オプション プルダウンメニューを使用して、以下のことができます。

- **大・小文字を区別** した検索の切り替え
- **フィルタパターン文法** を次のいずれかに設定

- **通常**
- **ワイルドカード**
- **正規表現**

プロパティウィジェット は、エントリ/リソースに関する有用な情報を表示します。 
プロパティウィジェットの有効化/無効化 ボタンを押すことで有効化/無効化することができます。有効にすると、 図 8.2 に示すようにブラウザパネルの下部にウィジェットが開きます。

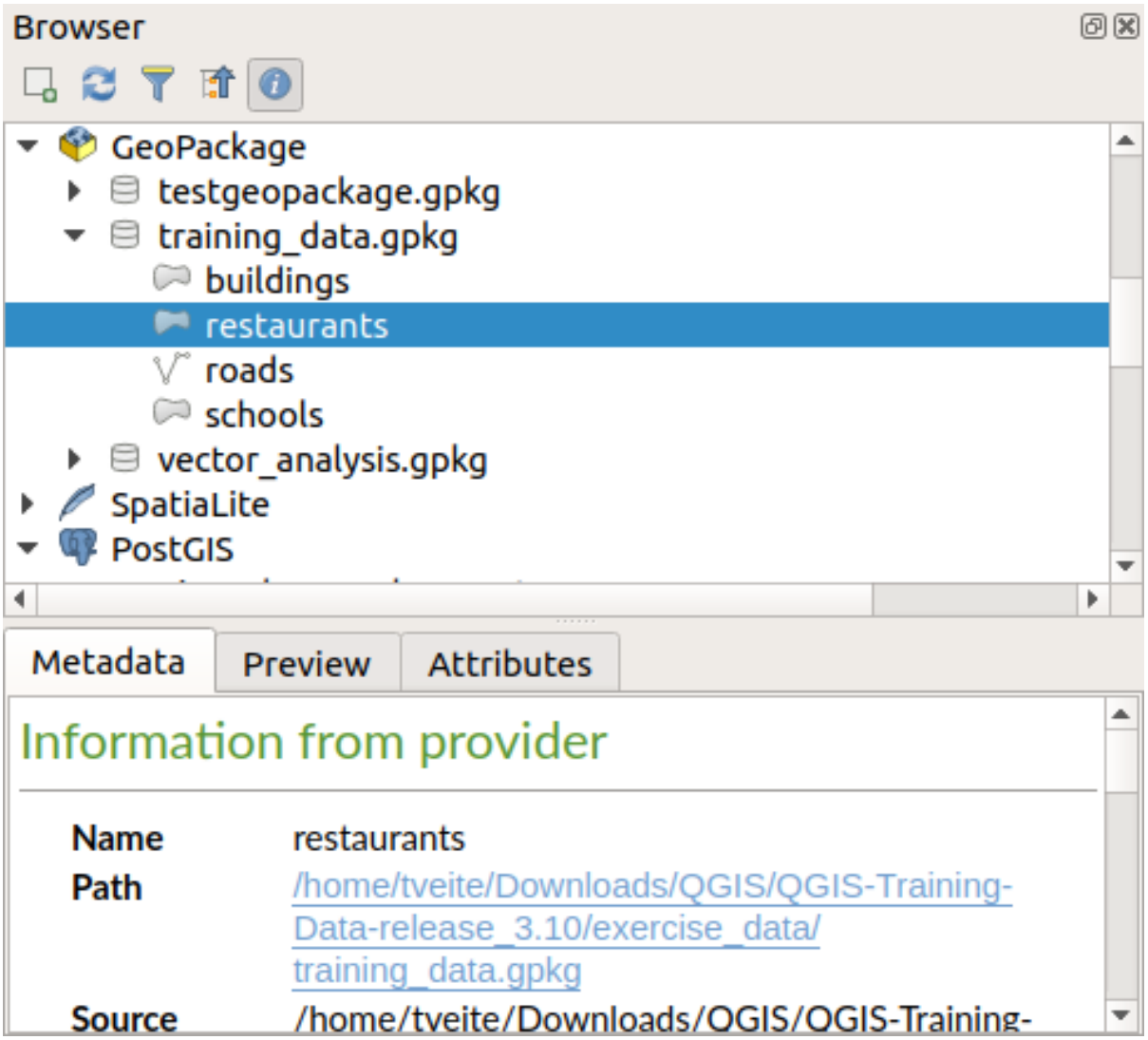


図 8.2: プロパティウィジェット

ビュー　パネル 内の **ブラウザ** *(2)* パネルを有効にすることで、2 つ目のブラウザパネルを開くことができます。2 つのブラウザパネルがあることは、ブラウザ階層のさまざまなブランチの深いところにあるリソース間でレイヤをコピーする際に役立ちます。

8.1 ブラウザから開くことができる/実行できるリソース

ブラウザパネルでは多くのことができます。

- ダブルクリック、マップキャンバス上へドラッグ、または（レイヤ選択後に）  選択したレイヤの追加 ボタンをクリックすることで、ベクタレイヤ、ラスタレイヤ、メッシュレイヤをマップに追加します。
- ダブルクリックもしくはマップキャンバスにドラッグして、Python スクリプト（プロセシングアルゴリズムを含む）を実行します。
- ダブルクリックもしくはマップキャンバスに上へドラッグして、プロセシングモデルを実行します。
- コンテキストメニューを使用して、QGIS プロジェクトから **シンボルを抽出**... します。
- デフォルトのアプリケーションでファイルを開きます（ コンテキストメニューの **外部で** *<file type>* **を開く**... ）。例：HTML ファイル、スプレッドシート、画像、PDF、テキストファイル等
- エントリのコピーを行います。
- レイヤの名前の変更や、レイヤの削除（複数可）（コンテキストメニューの **管理**　）
- ファイルエクスプローラを開き、そのファイルを直接選択する **ファイルに表示**

以下では、トップレベルのエントリ下で分類されたさまざまなリソースグループをリストしており、これに対するリソース固有のアクションを整理しています。

8.2 ブラウザパネルのトップレベルエントリ

8.2.1 お気に入り

よく使うファイルシステムの場所は、お気に入りとしてタグ付けすることができます。タグを付けたものはここに表示されます。

ホーム で説明する操作に加えて、コンテキストメニューでは **お気に入りの名前の変更**... と **お気に入りを削除** の操作を行うことができます。

8.2.2 空間ブックマーク

ここには空間ブックマークがあり、 **プロジェクト・ブックマーク** と **ユーザー・ブックマーク** に整理されています。

空間ブックマークのトップレベルのコンテキストメニューからは、ブックマークの作成（ **新規空間ブックマーク**... ） 、 **空間ブックマークマネージャを表示** 、 **空間ブックマークをインポート**... 、 **空間ブックマークをエクスポート**... を行うことができます。

ブックマークのグループエントリに対しては、 **空間ブックマークをエクスポート**... や、ブックマークの作成（ **新規空間ブックマーク**... ）、 **ブックマークグループ名を変更** 、 **ブックマークグループを削除** といった操作ができます。

ブックマークのエントリに対しては、 **ブックマークにズーム** 、 **空間ブックマークを編集...** 、 **ブックマークを削除** の操作がコンテキストメニューから行えます。

8.2.3 プロジェクトホーム

プロジェクトファイルが保存されると、 **プロジェクトホーム** のエントリが利用可能となります。これは、現在のプロジェクト内で使用されているデータやその他のコンテンツ（スクリプト、プロセシングモデル、テキスト等）を含むフォルダです。これは **ブラウザ** パネルに表示され、プロジェクトのデータやその他のファイルに素早くアクセスできるようになります。

これはデフォルトではプロジェクトファイルのあるフォルダですが、 **プロジェクト** → **プロパティ...** → **一般情報** → **プロジェクトのホーム** オプションや、ブラウザパネルの **プロジェクトホーム** アイテムを右クリックして **プロジェクトホームを設定...** から変更することができます。ホームフォルダのカスタマイズは、QGIS のプロジェクトがデータセットと一緒に組織の「プロジェクト」のルートフォルダに保存されていないような場合に特に便利です。

8.2.4 ドライブとファイルシステム

続く **ブラウザ** パネルのアイテムはお使いの OS によって異なり、ファイルシステムのトップレベルエントリに関係します。

これらは主に、以下のとおりです：

- **ホーム** フォルダ。現在のユーザーのホームフォルダを指します
- Unix ベースのマシンは、ルートフォルダ `/`
- 接続されたローカルドライブやネットワークドライブ。OS に依存し、直接表示される（例：`C:\`、`D:\`）場合もあれば、 `/Volumes` エントリを通じて表示される場合もあります。

これらのフォルダやドライブのコンテキストメニューからは、以下の操作が行えます：

- コンテンツの再読み込み
- **新規** のサブアイテムから、 **ディレクトリ** 、 *GeoPackage* 、または ESRI **シェープファイル** 形式データセットの作成
- ディレクトリの非表示（ **ブラウザから隠す** ）
- **色を設定** ：フォルダアイコンの色をカスタマイズすることで、複雑なフォルダ構造を素早くブラウズするのに役立ちます
- **スキャン** の有効化：
  - **変更を監視する** ：特定のディレクトリについて、モニタリングし自動更新するかどうかを手動で制御できます。この設定は、選択したディレクトリとそのサブディレクトリすべてに適用されます。これにより、ネットワークドライブに問題がないことが分かっている場合は監視を手動でオプトインしたり、その他の理由で監視したくない大きなディレクトリの監視を手動でオプトアウトしたりすることができます。デフォルトでは、リモートドライブやネットワークドライブは自動的には監視されません。

  - **このディレクトリの高速スキャン**

- ディレクトリをファイルマネージャで開く（**ディレクトリを開く**... ）
- ディレクトリをターミナルウィンドウで開く（**ターミナルで開く**... ）
- **プロパティ**... の確認や、親フォルダの **ディレクトリプロパティ**... の確認

8.2.5 データベースエントリ

ご利用の OS とインストールされているドライバによっては、QGIS で利用できるデータベースの種類が異なる場合があります。以下は、データセットツリーの各レベルのコンテキストメニューのさまざまなエントリのリストです。

レベル	コンテキストメニュー	データベースの種類					
		GeoPackage ([1])	SpatiaLite	PostGIS	SAP HANA	MS SQL Server	Oracle
トップメニュー	既存のデータベースへの **新規接続…** の作成	☑	☑	☑	☑	☑	☑
	データベースを作成…	☑	☑				
	ファイルに **接続を保存…**			☑	☑	☑	
	接続を読み込む…			☑	☑	☑	
接続 / データベース	接続の **再読み込み**			☑	☑	☑	
	接続の編集…			☑	☑	☑	
	接続を削除	☑	☑	☑	☑	☑	
	<database_1> **の削除**	☑	☑				

次のページに続く

表 8.1 – 前のページからの続き

	データベースの圧縮（*VACUUM*）	☑				
	新規スキーマ… の作成			☑	☑	☑
	新規テーブル… の作成	☑	☑	☑	☑	
	***SQL* を実行…**	☑	☑	☑	☑	
スキーマ	スキーマの **再読み込み**			☑	☑	☑
	スキーマ操作 スキーマ名の変更…			☑	☑	☑
	スキーマ操作 スキーマを削除…			☑	☑	☑
	新規テーブル… の作成			☑	☑	
	***SQL* を実行…**			☑	☑	
テーブル / レイヤ	**テーブル操作 テーブルの名前を変更…**			☑	☑	☑
	テーブル操作 テーブルを全行削除…			☑		☑
	***SQL* を実行…**	☑	☑	☑		

次のページに続く

表 8.1 – 前のページからの続き

	レイヤエクスポート ファイルへ…	☑	☑	☑	☑	☑
	管理 <layer_nam のレイヤ名変更…	☑	☑			
	管理 レイヤ <layer_nam を削除する…	☑	☑	☑	☑	☑
	管理 選択したレイヤを削除	☑	☑	☑	☑	☑
	管理 レイヤをプロジェクトに追加する	☑	☑	☑	☑	☑
	管理 選択されたレイヤをプロジェクトに追加する	☑	☑	☑	☑	☑
	レイヤのプロパティ… ダイアログを開く	☑	☑	☑	☑	☑
	ファイルのプロパティ… ダイアログを開く	☑				
属性テーブル	**新規属性を追加…**	☑	☑	☑	☑	
属性	**属性を削除…**	☑	☑	☑	☑	

8.2.6 タイルと Web サービス

レベル	コンテキストメニュー	サービスの種類						
		WMS / WMTS	Vector Tiles	XYZ Tiles	WCS	WFS / OGC API - Features	ArcGIS REST Servers	GeoNode
トップメニュー	**新規接続…** の作成	✓		✓	✓	✓	✓	✓
	新規一般接続… の作成		✓					
	新規 *ArcGIS Vector Tile Service* **接続…** の作成		✓					
接続	ファイルに **接続を保存…**	✓	✓	✓	✓	✓	✓	✓
	接続を読み込む…	✓	✓	✓	✓	✓	✓	✓
	接続の **再読み込み**	✓		✓	✓	✓	✓	✓
	接続の編集…	✓	✓	✓	✓	✓	✓	✓
	接続を削除	✓	✓	✓	✓	✓	✓	✓
	ウェブブラウザで **サービス情報を見る**						✓	
テーブル / レイヤ	**レイヤエクスポート　ファイルへ…**	✓		✓	✓	✓	✓	✓
	レイヤをプロジェクトに追加する	✓	✓	✓	✓	✓	✓	✓
	レイヤのプロパティ… ダイアログを開く	✓	✓	✓	✓	✓	✓	✓
	ウェブブラウザで **サービス情報を見る**						✓	

[1] GDAL がサポートしている `ベクタファイルフォーマット <https://gdal.org/drivers/vector/index.html>`_（ESRI File Geodatabase、FlatGeobuf、GeoParquet、NetCDF、... など）についても、互換性があれば異なるエントリが利用できるかもしれません。

8.3 リソース

- プロジェクトファイル：QGIS プロジェクトのコンテキストメニューでは、以下の操作があります。
  - 開く（ **プロジェクトを開く** ）
  - シンボルを抽出する（ **シンボルを抽出**... ） — スタイルマネージャを開き、シンボルを XML ファイルに追加したり、シンボルをデフォルトのスタイルに追加したり、PNG や SVG としてエクスポートしたりすることができます。
  - プロパティを調べる（ **ファイルプロパティ**... ）

  プロジェクトファイルを展開して、プロジェクトのレイヤを確認できます。レイヤのコンテキストメニューは、ブラウザ内の他のエントリのレイヤにおけるアクションと同じです。
- QGIS レイヤ定義ファイル（QLR）：コンテキストメニューからは以下の操作が可能です。
  - ファイルに保存する（ **レイヤエクスポート　ファイルへ** ）
  - プロジェクトへ追加する（ **レイヤをプロジェクトに追加する** ）
  - プロパティを調べる（ **レイヤのプロパティ**... ）
- プロセシングモデル（.model3）：コンテキストメニューからは以下の操作が可能です。
  - **モデルを実行**...
  - **モデルの編集**...
- QGIS 印刷レイアウトテンプレート（QPT）：コンテキストメニューからは以下の操作が可能です。
  - **テンプレートから新規作成**
- Python スクリプト（.py）：コンテキストメニューからは以下の操作が可能です。
  - **スクリプトの実行**...
  - **外部エディタで開く**
- 認識されるラスタ形式ファイル：コンテキストメニューからは以下の操作が可能です。
  - <*dataset name*> **を削除**
  - ファイルに保存する（ **レイヤエクスポート　ファイルへ** ）
  - プロジェクトへ追加する（ **レイヤをプロジェクトに追加する** ）
  - プロパティを調べる（ **レイヤのプロパティ**... 、 **ファイルプロパティ**... ）

  一部の形式は、 **外部で** <*file type*> **を開く**... こともできます。
- 認識されるベクタ形式ファイル：コンテキストメニューからは以下の操作が可能です。
  - <*dataset name*> **を削除**
  - ファイルに保存する（ **レイヤエクスポート　ファイルへ** ）
  - プロジェクトへ追加する（ **レイヤをプロジェクトに追加する** ）
  - プロパティを調べる（ **レイヤのプロパティ**... 、 **ファイルプロパティ**... ）

一部の形式は、 **外部で** <*file type*> **を開く**... こともできます。

第9章　QGISの設定

QGIS では高度な設定が可能です。 **設定** メニューを通じて、以下のさまざまなツールを提供します。

- **スタイルマネージャ**... : は **シンボル、スタイル、カラーランプ** を作成し、管理します。
- **カスタム投影法**...: では独自の **座標参照系** を作成します。
- **キーボードショートカット**...: は自分用の **キーボードショートカット** のセットを定義します。また、それらは各 QGIS セッション中に **プロジェクトのプロパティ** （ **プロジェクト** メニューからアクセス可能）によって上書きできます。
- **インタフェースのカスタマイズ**... では **アプリケーションインタフェース** の設定を行い、不要なダイアログやツールを非表示にできます。
- **オプション**...: は、ソフトウェアのさまざまな領域に適用するグローバル **オプション** を設定します。これらの設定は、アクティブな **ユーザープロファイル** 設定に保存され、このプロファイルで新しいプロジェクトを開く場合にはデフォルトで適用されます。

9.1 オプション

QGIS のいくつかの基本オプションは、**オプション** ダイアログを使用して選択できます。メニューオプション **設定**　 **オプション** を選択してください。必要に応じてオプションを変更できます。 オプションが有効となるためには QGIS の再起動が必要なものもあります。

以下で、オプションをカスタマイズできる画面をタブごとに説明します。

注釈: プラグインの設定はオプションダイアログに埋め込み可能

以下ではコア設定のみを紹介しますが、 **インストールされたプラグイン** が独自のオプションを標準のオプションダイアログ内に実装することで、このリストを拡張できることに注意してください。これにより、それぞれのプラグインが独自の設定ダイアログを持ち、それらのためだけに余分なメニュー項目を持つことを避けられます...

9.1.1 一般情報の設定

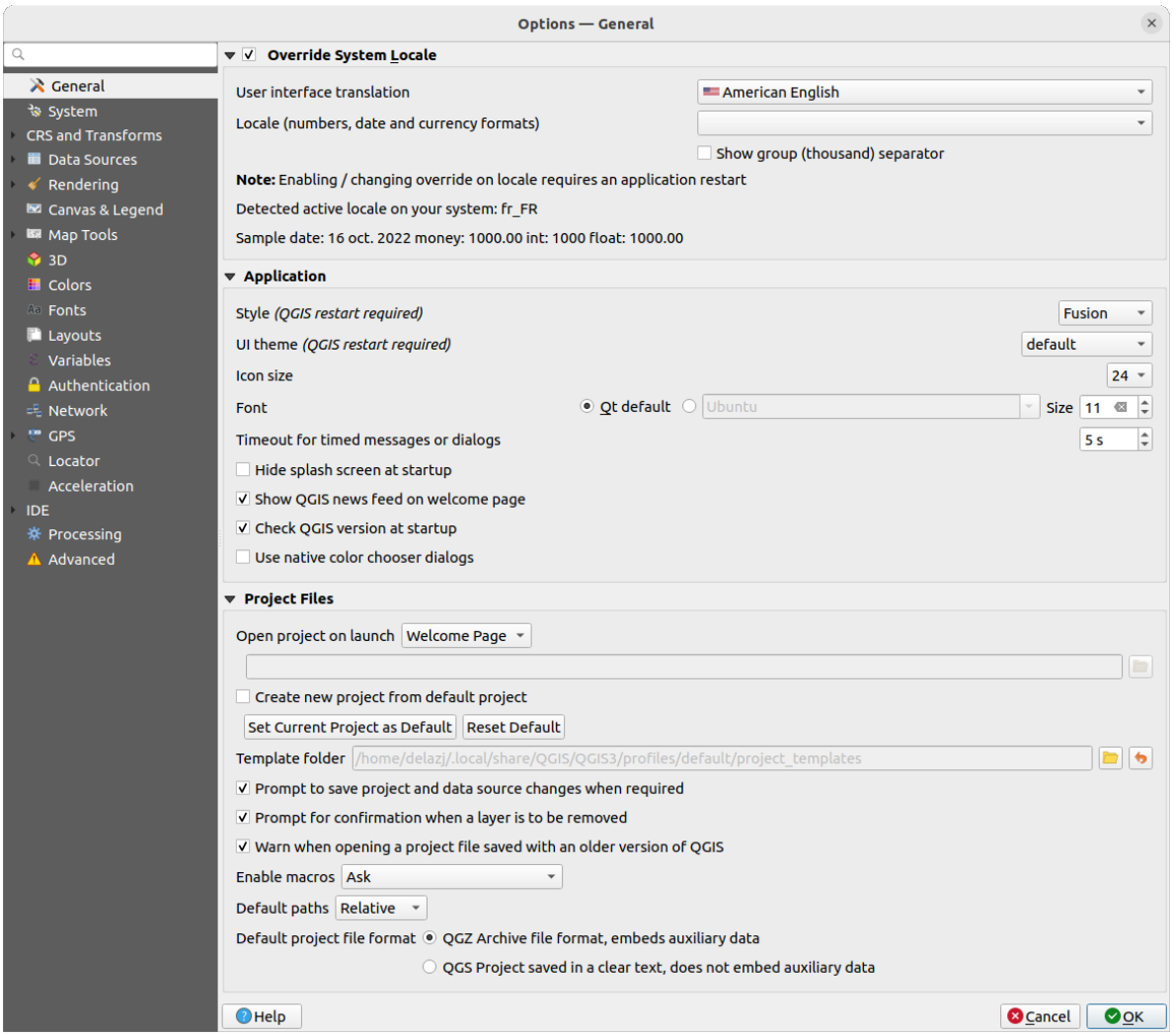


図 9.1: 一般情報の設定

システムロケールを上書き

デフォルトでは、QGIS の言語の設定と数字の扱いはオペレーティングシステムの設定に依存しています。このグループを有効にすると、動作をカスタマイズすることができます。

- **ユーザーインターフェイスの言語** で、GUI に適用する言語を選択する
- **ロケール（数字、日付、通貨の形式）** で、日付や数字の入力・表示形式を選択する
- **数字の *3* 桁区切り記号を表示**

選択された設定の概要とその設定がどのように解釈されるかについては、フレームの下部に表示されます。

アプリケーション

- **スタイル（*QGIS* の再起動が必要）** の選択、つまり、ウィジェットの外観やダイアログ内の位置を設定します。可能な値は、オペレーティングシステムによって異なります。

- *UI* **テーマ（*QGIS* の再起動が必要）**  の設定。'default'、 'Night Mapping'、または 'Blend of Gray' を使用できます。
- **アイコンサイズ**  を定義します
- **フォント** と **サイズ** を定義します。フォントは  *Qt* **のデフォルト** か、ユーザ定義のフォントにすることができます。
- **メッセージやダイアログのタイムアウト設定** を変更します
-  **起動時にスプラッシュスクリーンを表示しない**
-  *QGIS* **ニュースを起動ページに表示** ：QGIS ニュースフィードを起動ページに表示します。これによりプロジェクトのニュース（ユーザー/開発者の会議の日付と概要、コミュニティ調査、リリースのお知らせ、さまざまなヒント...）を知ることができます
-  **起動時に *QGIS* バージョンを確認する** は、新しいバージョンがリリースされたときに知らせてくれます
-  **ネイティブの色選択ダイアログを使用する** （ **カラーセレクタ** を参照）

プロジェクトファイル

- **起動時に開くプロジェクト**
  - 「ウェルカムページ」（デフォルト）は、 "ニュース ” フィードやプロジェクトテンプレート、**ユーザープロファイル** の最近使用したプロジェクト（とそのサムネイル画像）を表示します。デフォルトではプロジェクトは開かれません。
  - 「新規」: は、デフォルトのテンプレートに基づいて新しいプロジェクトを開きます。
  - 「最近」: は、最後に保存したプロジェクトを再度開きます。
  - 「テンプレートを指定」: は、特定のプロジェクトファイルを開きます。 ... ボタンを押して、デフォルトで使用するプロジェクトを指定します。
-  **既定のプロジェクトから新プロジェクトを作成する** では、 **現在のプロジェクトを既定のプロジェクトとして設定する** または **デフォルトにリセット** ボタンをクリックすることができます。ファイルをブラウズして、ユーザー定義のプロジェクトテンプレートがあるディレクトリを指定することもできます。これは、 **プロジェクト　テンプレートから新規作成** に追加されます。最初に  **既定のプロジェクトから新プロジェクトを作成する** をアクティブにしてから、プロジェクトテンプレートフォルダにプロジェクトを保存してください。
-  **必要なときにプロジェクトおよびデータソースの変更を保存するか尋ねる** は、変更が失われないよう、必要な時に尋ねます。
-  **レイヤが削除されるときに確認プロンプトを表示する**
-  **古いバージョンの *QGIS* で保存されたプロジェクトファイルを開く時に警告する** 古いバージョンの QGIS で作成されたプロジェクトは常に開くことができますが、一旦プロジェクトを保存してしまうと、古いバージョンの QGIS では開けなくなることがあります。古いバージョンでは使用できない機能があるためです。
- **マクロを有効にする**  このオプションは、プロジェクトのイベントのアクションを実行するために書かれたマクロを扱うためにあります。このオプションは、「利用しない」、「確認する」、「この

セッションのみ」、「このセッションは無効」、「常に許可（非推奨）」の中で選択できます。

- **デフォルトパス** 新しいプロジェクトで使用するファイルやレイヤへのパスの保存方法を「絶対パス」とするか、プロジェクトに対する「相対パス」とするかを定義します。この設定は、プロジェクトレベルで上書きできます。
- **デフォルトのプロジェクトファイル形式**
  -  *QGZ Archive* **ファイル（補助データ付き）**（**補助データ** 参照）
  -  *QGS Project* **ファイル（テキスト形式、補助データなし）**：補助データはプロジェクトファイルとは別に、`.qgd` ファイルに保存されます。

9.1.2 システムの設定

SVG パス

Scalable Vector Graphic（*SVG*）**シンボルを探すパス** を追加・削除します。この SVG ファイルは地物のシンボルやラベル、印刷レイアウトの装飾に利用できます。

QGIS のパスで SVG ファイルを参照する別の方法として、**リモート / 埋め込みファイルセレクタ** も参照してください。

プラグインパス

追加の *C++***プラグインライブラリを探すパス** を追加・削除します。

ドキュメントのパス

QGIS **のヘルプを探すパス** を追加・削除します。デフォルトでは、使用されているバージョンに対応する公式オンラインユーザーマニュアルへのリンクが追加されています。ただし、他のリンクを追加することもできます。ダイアログの **ヘルプ** ボタンをクリックするたびに一番上のリンクがチェックされ、対応するページが見つからない場合は次のリンクが順に試されるので、追加した他のリンクに上から下へと優先順位をつけられます。

注釈: ドキュメントは QGIS 長期リリース（LTR）に対してのみバージョン管理され翻訳されています。つまり、通常のリリース（QGIS 3.0 など）を実行している場合、ヘルプボタンはデフォルトで次の LTR マニュアルページ（3.4 LTR）を開くため、新しいリリース（3.2 および 3.4　　）の機能の説明が含まれる場合があります。 LTR ドキュメントが利用できない場合は、新しいバージョンと開発バージョンの機能が記載された *testing* ドキュメントが使用されます。

設定

ユーザーインターフェイスを既定の設定にリセット（*QGIS* **の再起動が必要）** は、何か **カスタマイズ** をした後で、設定をリセットする際に役立ちます。

環境

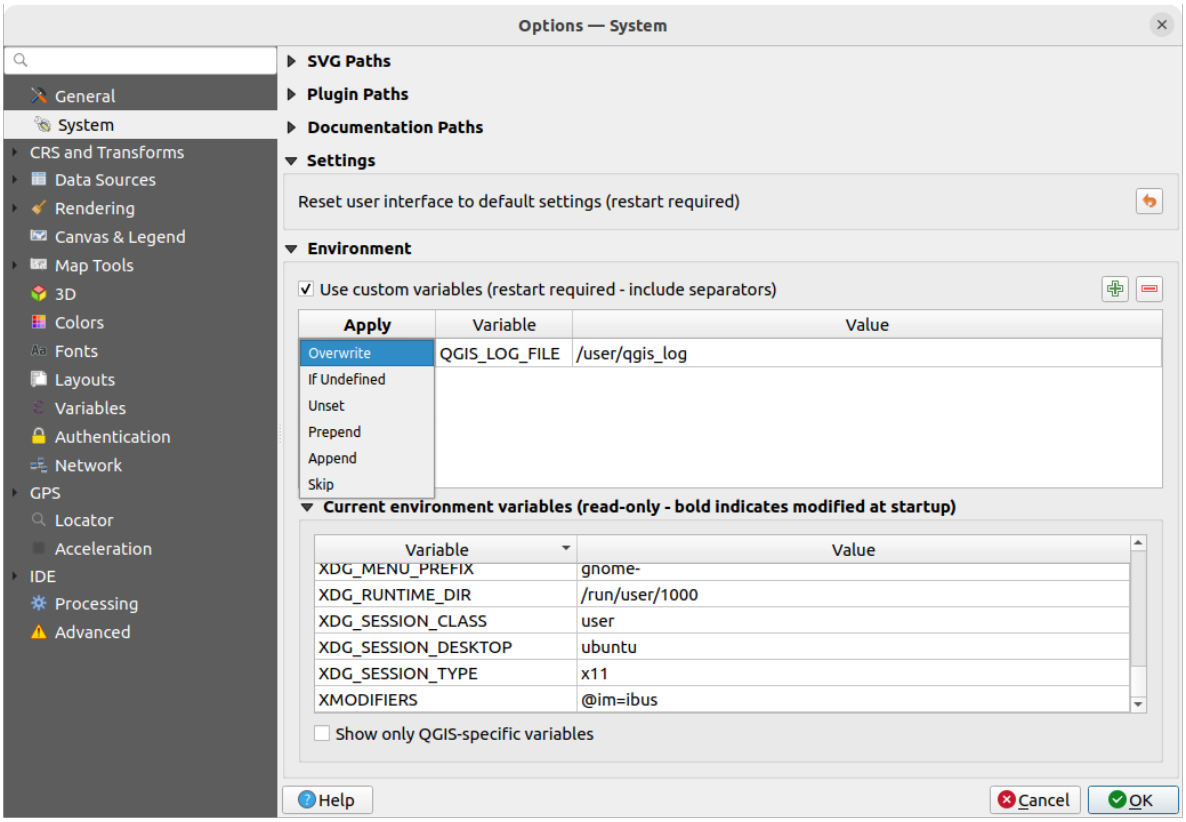


図 9.2: システム環境変数

環境 グループでは、システム環境変数を表示し、多くの設定を行うことができます。これは Mac など、GUI アプリケーションが必ずしもユーザのシェル環境を継承しないプラットフォームで役立ちます。また、Processing ツールボックス（SAGA、GRASS など）によって制御される外部ツールセットの環境変数を設定および表示したり、ソースコードの特定のセクションのデバッグ出力をオンにしたりする場合にも役立ちます。

カスタム変数を用いる（区切り文字を含む－ *QGIS* の再起動が必要） をチェックすると、環境変数を 新規変数を追加 と 変数を削除 することができます。新しい各項目で、**変数** 名、**値** および **適用** 方法を設定することができます。

- **上書き**: 存在していた変数の値を置き換えます
- **未定義の場合**: もし（OS やアプリケーションレベルなどの）高いレベルでまだ定義されていなければ、この値をその変数に使います
- **設定取り消し**: 環境からこの変数を削除します（**値** パラメータは使われません）
- **先頭に追加**: 値をこの変数にセットされている値の先頭に追加します
- **追加**: 値をこの変数にセットされている値に追加します
- **スキップ**: この項目は将来の参照用にリストに保存されますが使われません

既に定義されている環境変数は **現在の環境変数** に表示され、 *QGIS* **固有の変数のみ表示** を有効にしてフィルタリングすることが可能です。

9.1.3 座標参照系と変換

注釈: QGIS がレイヤの投影をどのように扱っているかについての詳細は、専用のセクションである **投影法の利用方法** を読んでください。

CRS の扱い

CRS **の扱い** タブでは新しいプロジェクト又はレイヤにどの CRS を使うかを設定することができます。

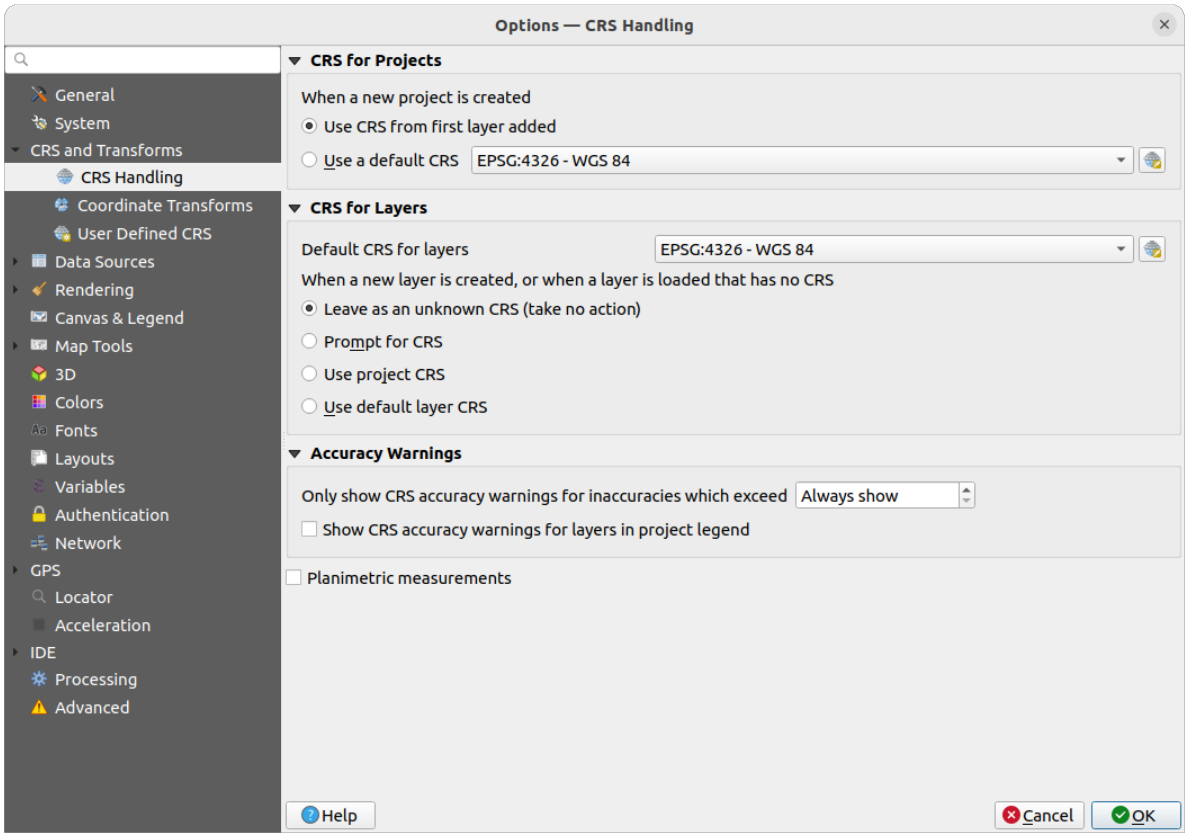


図 9.3: CRS の設定

プロジェクトの座標参照系 (CRS)

新しいプロジェクトの CRS を自動的に設定するオプションがあります。

- **最初のレイヤの *CRS* を使う**：プロジェクトの CRS は、ロードされた最初のレイヤの CRS に設定されます
- **デフォルトの *CRS* を使う**：事前に選択された CRS がデフォルトで新しいプロジェクトに適用され、プロジェクトにレイヤを追加しても変更されません。

この選択内容は保存され、以降の QGIS セッションでも使われます。プロジェクトの座標参照系は、**プロジェクト　プロパティ...　座標参照系（*CRS*）** タブから上書きすることができます。

レイヤの CRS

レイヤのデフォルト *CRS* は、レイヤを作成した時のデフォルトの CRS を選択します

新しいレイヤを作成する時や、CRS のないレイヤが読み込まれた時に実行するアクションを定義することもできます。

- **未知の** *CRS* **のまま**（何もしない）
- *CRS* **ダイアログを表示**
- **プロジェクトの** *CRS* **を使用**
- **デフォルト** *CRS* **を使う**

精度に関する警告

指定した距離の **精度警告の閾値**：データセットを明示的に作成または変更する際に、精度の低いデータムアンサンブルに基づく CRS を選択したときに発生します。デフォルトでは、不正確な場合は常に警告を表示するようになっています。少なくとも PROJ 8.0 を使用する QGIS バージョンが必要です。

CRS **精度警告を表示**： チェックした場合、精度に問題がある CRS（座標エポックが利用できない動的 CRS、またはユーザーが設定した制限を超える固有の不正確さを持つデータムアンサンブルに基づく CRS）を持つレイヤは、低精度レイヤであることを示す 警告アイコン が **レイヤ** パネル内に表示されます。

これは、エンジニアリングや BIM、アセットマネジメントなど、メートル級 / サブメートル級の不正確さが非常に危険または高価になる可能性のある分野での使用を想定しています。

平面計測: 新しくプロジェクトを作成した時の **平面計測** プロパティのデフォルトを設定します。

座標変換

座標変換 タブでは、レイヤをプロジェクトに読み込むときやレイヤを再投影するときに適用する座標変換や操作を設定できます。

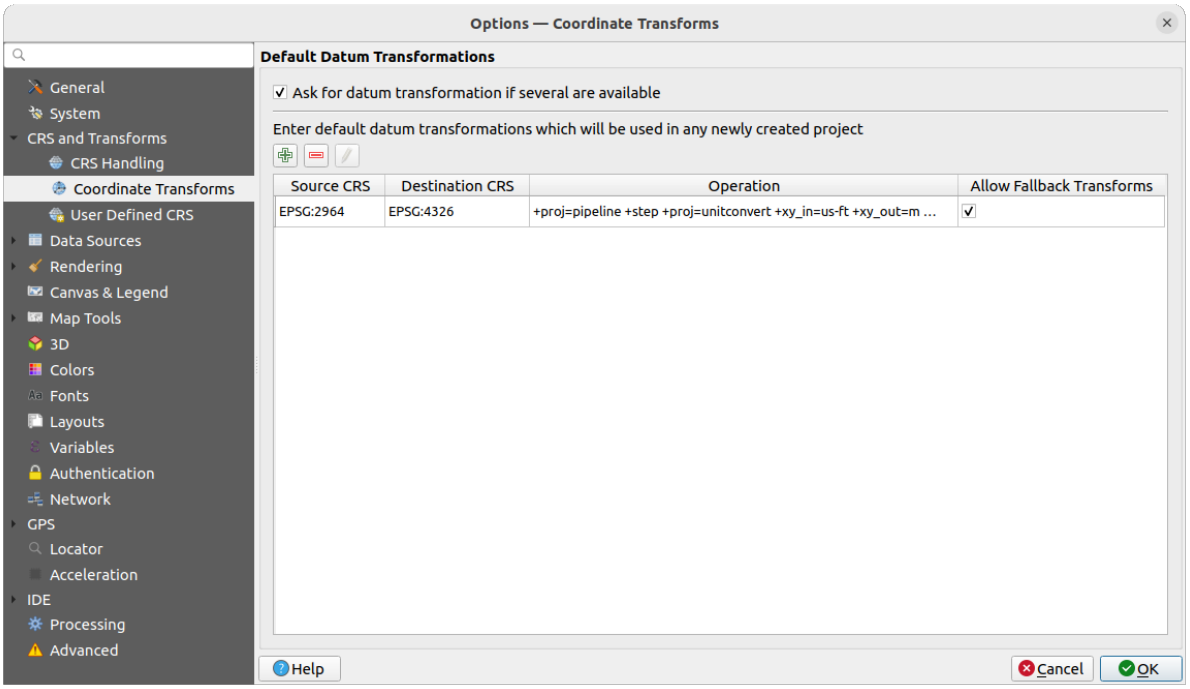


図 9.4: 変換の設定

デフォルトの測地系変換

レイヤを別の CRS に再投影するかどうかを制御します：

- QGIS のデフォルト変換設定を使用して自動的に処理する
- 以下のカスタム設定でユーザがより細かく制御する
  - **測地系変換が複数利用可能な場合は尋ねる**
  - デフォルトで適用する測地系変換の事前定義リスト。詳細は **測地系変換** を参照してください。

変換を 追加、 削除 または 編集 することができ、これらは新しく作成されたプロジェクトで使用されます。

ユーザ定義 CRS

ユーザー定義 *CRS* タブでは、WKT または Proj 文字列形式に準拠したカスタム CRS を定義することができます。

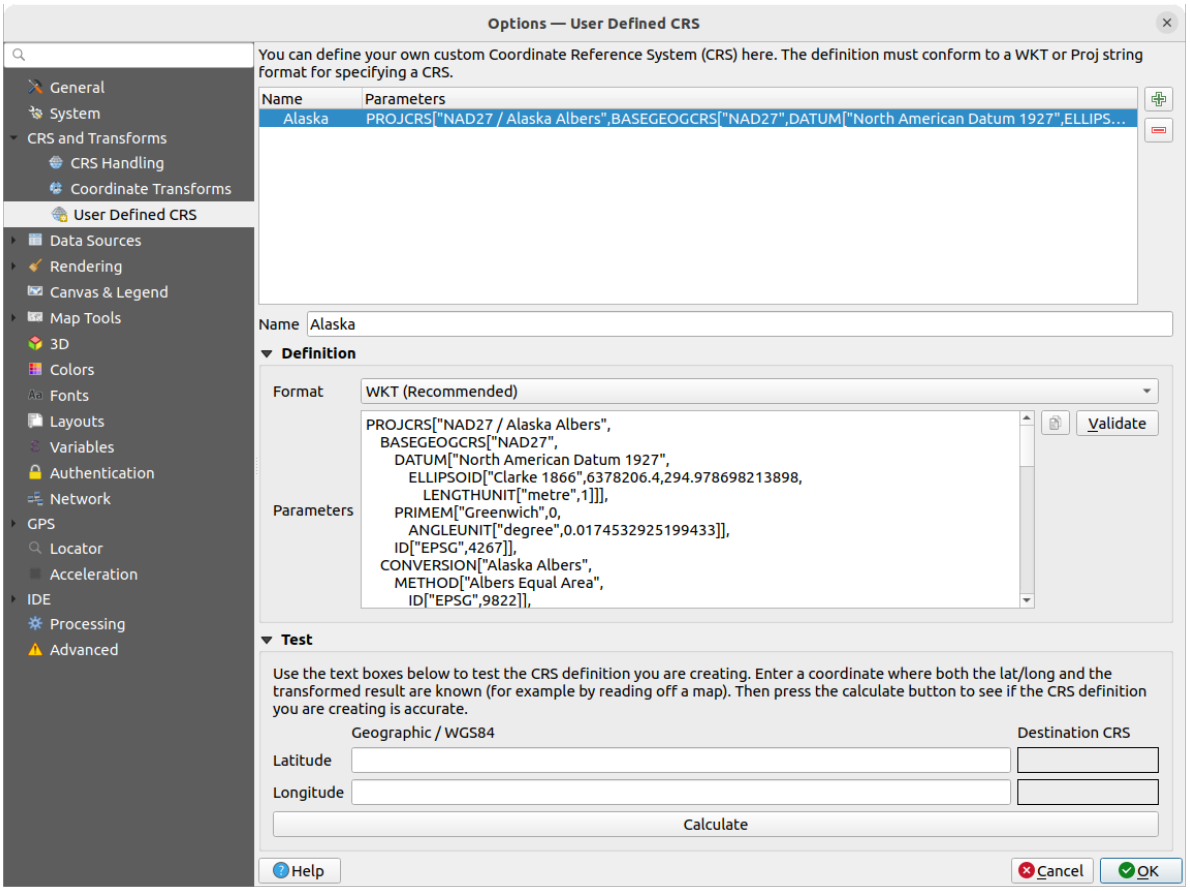


図 9.5: ユーザ定義 CRS

名前 を設定し、 CRS を追加 を使います。既存のものを削除したい場合は、 CRS を削除 を使います。

定義

- **形式**
  - WKT（推奨）
  - Proj 文字列（非推奨）
- **パラメータ**
  - 既存の CRS からパラメータをコピーする。
  - **検証** 式が正しいかテストします。

テスト

ここでは、作成した CRS の定義を緯度と経度でテストすることができます。既知の座標を使用して、定義が正確であるかどうかを制御します。

9.1.4 データソースの設定

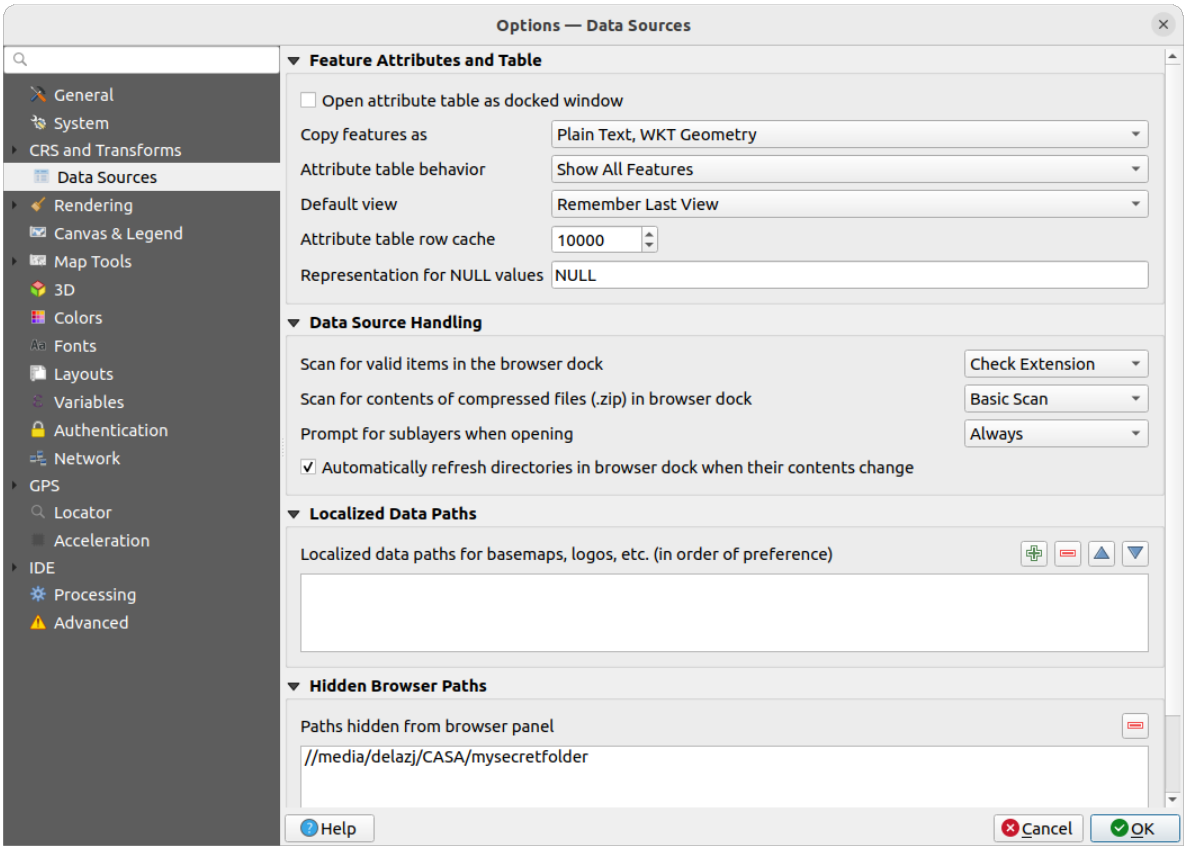


図 9.6: データソースの設定

地物属性とテーブル

- **ドックウィンドウとして属性テーブルを開く**
- **地物をコピー** は、地物をコピーして他のアプリケーションに貼り付けるときの形式として、「プレーンテキスト（ジオメトリなし）」、「プレーンテキスト（WKT ジオメトリ）」もしくは、「GeoJSON」のいずれかを設定します。
- **属性テーブルの動作** ：属性テーブルを開いた時のフィルタを設定します。「すべての地物を表示」、「選択した地物を表示」、「地図上に表示されている地物を表示」の 3 つうちのいずれかです。
- **デフォルト表示形式**：属性テーブルを開いた時の表示モードを定義します。「直前の表示形式」、「テーブル表示」、または「フォーム表示」のいずれかです。
- **属性テーブル行キャッシュ** 1,00 。この行キャッシュは、最後にロードされた属性を N 行分保存することで、属性テーブルでの作業をより速くできるようにします。キャッシュは属性テーブルを閉じるときに削除されます。
- *NULL* **値の表示方法** 。NULL 値を含んだデータフィールドの値を定義できます。

Tip: 巨大な属性テーブルを開く際の動作を改善する

大量のレコードを含むレイヤを操作する場合、ダイアログがレイヤ内のすべての行をリクエストすると、属性テーブルを開くのが遅くなることがあります。 **属性テーブルの動作** を **地図上に表示されている地物を表示** に設定すると、QGIS はテーブルを開くときに現在の地図キャンバス内にある地物だけをリクエストするため、データの読み込みが早くなります。

この属性テーブルインスタンスのデータは、開いたキャンバスの範囲に常に関連付けられます。つまり、そのテーブル内の **全地物を表示** を選択すると新しい地物は表示されません。ただし、キャンバス範囲を変更して、属性テーブルの **地図上に表示されている地物を表示** オプションを選択すると、表示されている地物のセットを更新できます。

データソースの操作

- **ブラウザドック内で有効なアイテムをスキャンする** 。「拡張子をチェック」と「ファイルの内容をチェック」のどちらかを選択できます。
- **ブラウザドック内で圧縮ファイル（*.zip*）にあるコンテンツをスキャンする** は、圧縮ファイルの情報を確認する際に、ブラウザパネル下部にプロパティウィジェット情報をどの程度詳細に表示するかを定義します。 「No」、「ベーシックスキャン」、「フルスキャン」の選択肢があります。
- **ラスタサブレイヤを開くときにプロンプトを表示** 。ラスタの中にはサブレイヤをサポートするものがあり、それらは GDAL ではサブデータセットと呼ばれています。例えば、netCDF ファイルです。多数の netCDF 変数がある場合、GDAL はすべての変数をサブデータセットとして認識します。このオプションは、サブレイヤを持つファイルが開かれたときに、サブレイヤをどう扱うかを制御することができます。以下の選択肢があります：
  - 「常に」：（サブレイヤが存在する場合には）毎回、確認プロンプトを表示します
  - 「必要な場合」：レイヤがバンドは無いがサブレイヤを持っている場合にプロンプトを表示します
  - 「利用しない」:プロンプトを表示せず、何もロードしません
  - 「すべて読み込む」:プロンプトは表示しませんが、すべてのサブレイヤをロードします
- **内容が変更された時、自動でブラウザのディレクトリを更新** ：デフォルトで行われる **ブラウザ** パネル内のディレクトリの監視を、実行しないように手動で設定できます（例：ネットワーク遅延による潜在的な速度低下を回避するため）。

ローカルデータのパス

任意の種類のファイルベースのデータソースに対して、ローカライズされたパスを使用することが可能です。ローカライズされたパスとは、データソースの位置を抽象化するために使用されるパスのリストです。例えば、`C:\my_maps` がローカルデータのパスのリストにあるとすると、`C:\my_maps\my_country\ortho.tif` をデータソースとして持つレイヤは、 `localized:my_country\ortho.tif` としてプロジェクト内に保存されます。

パスは、優先順位の高い順にリストアップされています。言い換えると、QGIS はまず最初のパスでファイルを探し、次に 2 番目のパスで、といった具合にファイルを探します。

ブラウザに表示しないパス

このウィジェットは **ブラウザパネル** から非表示にすることを選択したすべてのフォルダをリストアップします。リストからフォルダパスを削除すると、 **ブラウザ** パネルで表示できるようになります。

GDAL の設定

GDAL は地理空間データのためのデータ変換ライブラリで、多数のベクタ形式やラスタ形式をサポートしています。GDAL はデータの読み込みと、（大抵の場合）これらの形式でデータを書き出すためのドライバを提供しています。 *GDAL* タブではラスタ形式およびベクタ形式用のドライバと、その機能を公開しています。

GDAL のラスタドライバとベクタドライバ

複数の GDAL ドライバが利用可能な場合があるため、 **ラスタドライバ** と **ベクタドライバ** のタブでは、読み書きにどの GDAL ドライバを有効にするかを定義できます。

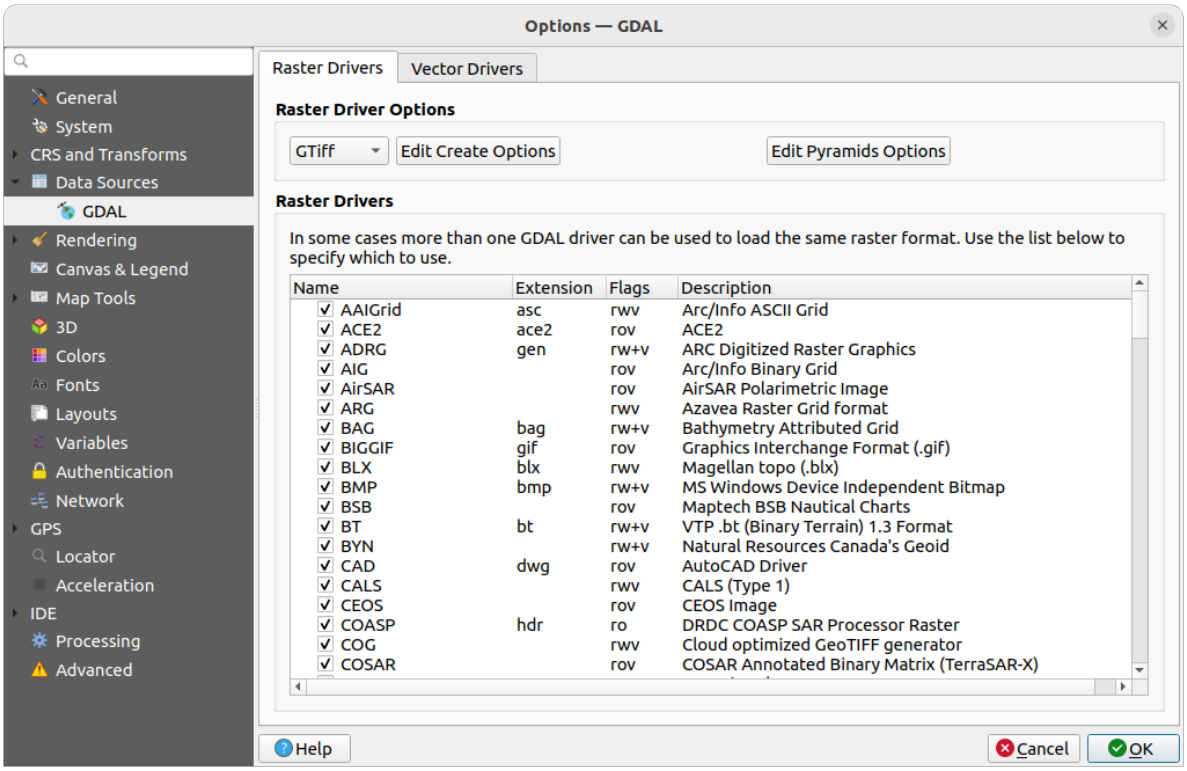


図 9.7: GDAL の設定 — ラスタドライバ

Tip: 読み取り・書き込みアクセスが可能（ `rw+(v)` ）なラスタドライバをダブルクリックすると、 **作成オプションの編集** ダイアログが開き、作成オプションをカスタマイズできます。

ラスタドライバオプション

このフレームでは、読み取りおよび書き込みアクセスをサポートするラスタドライバの動作をカスタマイズする方法を提供します。

- **作成オプションを編集**: ファイル変換の際に使うさまざまなプロファイルを編集または追加することができます。すなわち、ラスタファイルを出力する際に使用するパラメータ（圧縮の種類と圧縮レベル、ブロックサイズ、全体図、測色、アルファ... ）の事前に定義された組み合わせのセットを編集・追加できます。パラメータはドライバに依存します。

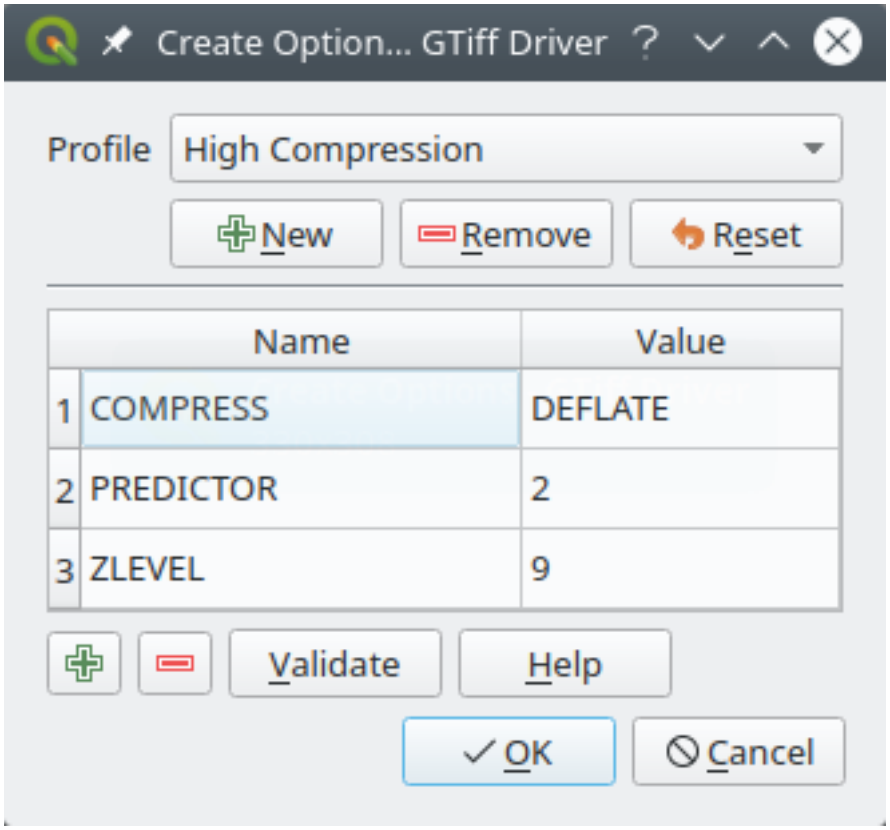


図 9.8: オプションプロファイルの作成サンプル（GeoTiff ドライバ）

ダイアログの上部には現在のプロファイルがリストで表示され、新しいプロファイルを追加したり、削除したりできます。プロファイルを変更した場合に、デフォルトのパラメータにリセットすることもできます。一部のドライバ（GeoTiff など）には、プロファイルのサンプルがあります。

ダイアログの下部では、以下の操作ができます：

- ボタンを押すと、パラメータの名前と値を入力できる行を追加します
- ボタンを押すと、選択したパラメータを削除します
- **検証** ボタンをクリックすると、指定したフォーマットに対して入力した作成オプションが有効であるかを確認します
- **ヘルプ** ボタンを押して使用するパラメータを見つけるか、GDAL raster drivers documentation を参照してください。

- **ピラミッドオプションを編集**

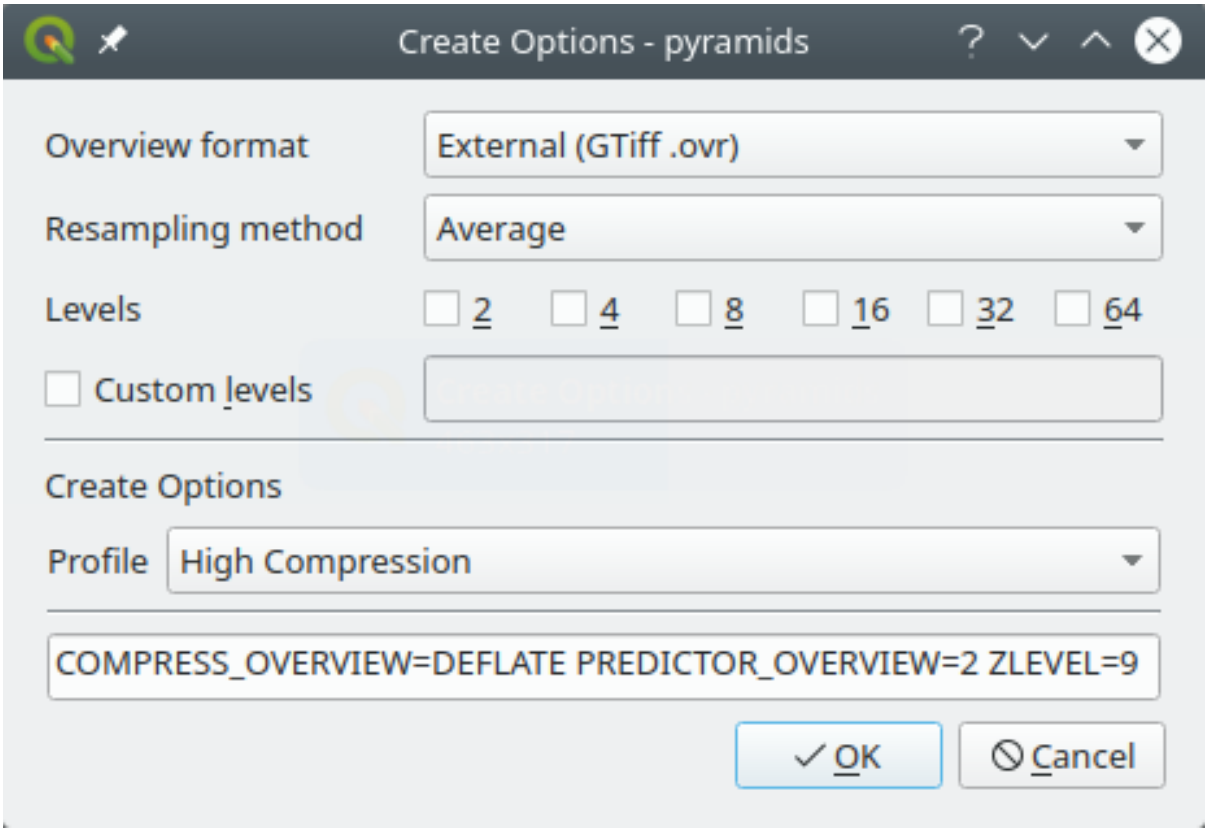


図 9.9: ピラミッドプロファイルのサンプル

9.1.5 レンダリングの設定

レンダリング タブは、マップキャンバスのレイヤのレンダリングを制御するための設定を提供します。

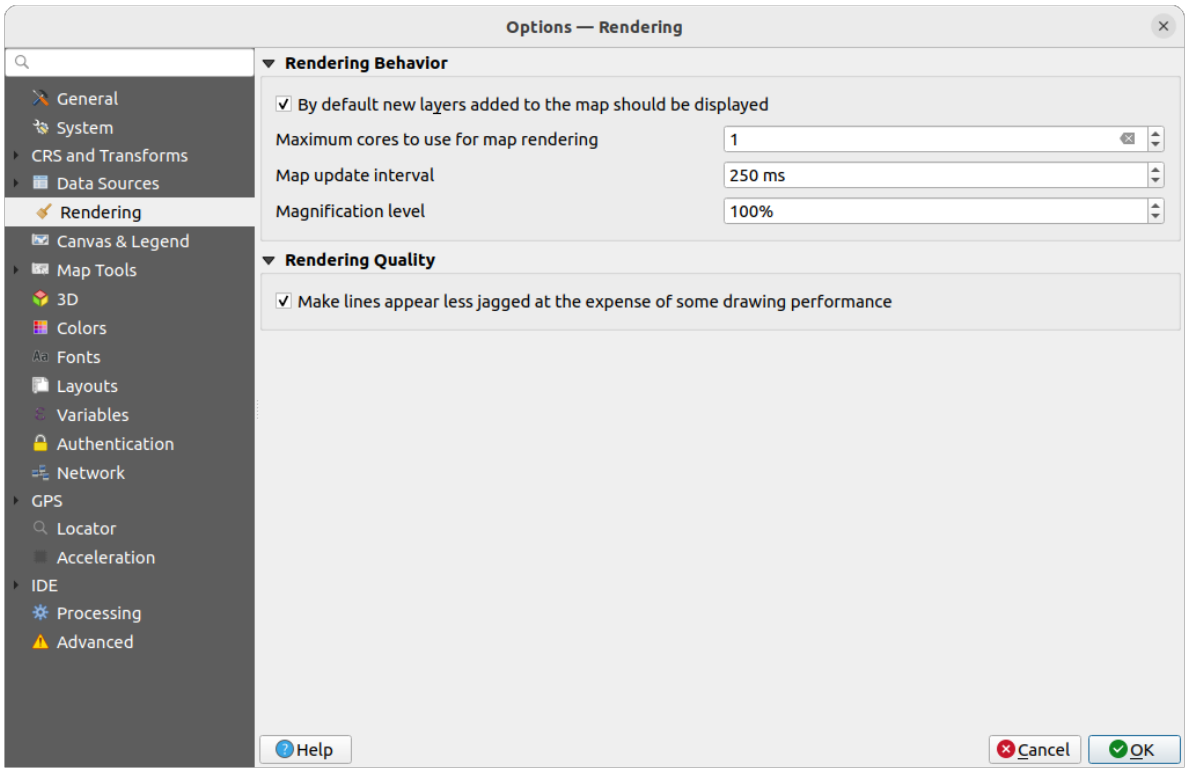


図 9.10: レンダリングの設定

レンダリング動作

- **地図に追加された新規レイヤはデフォルトで表示**：このオプションのチェックを外すと、新しいレイヤが読み込まれたときにキャンバスにレンダリングされてプロセスが遅くなるのを避けられるため、複数のレイヤを読み込む際に便利です。
- **レンダリングに使う** *CPU* **コアの最大数** を設定する
- マップキャンバスはバックグラウンドで別の画像にレンダリングされ、**更新間隔**（デフォルトは 250 ミリ秒）ごとに、この（オフスクリーン）画像の内容がオンスクリーンの表示を更新するために使用されます。しかし、レンダリングがこの時間よりも早く終了した場合は、即座に表示されます。
- **拡大レベル**（**拡大率** を参照）

レンダリング品質

- **線のジャギーを目立たなくする（描画速度が若干低下します）**

ベクタレンダリングの設定

ベクタ タブにはベクタレイヤをレンダリングするための特定の設定があります。

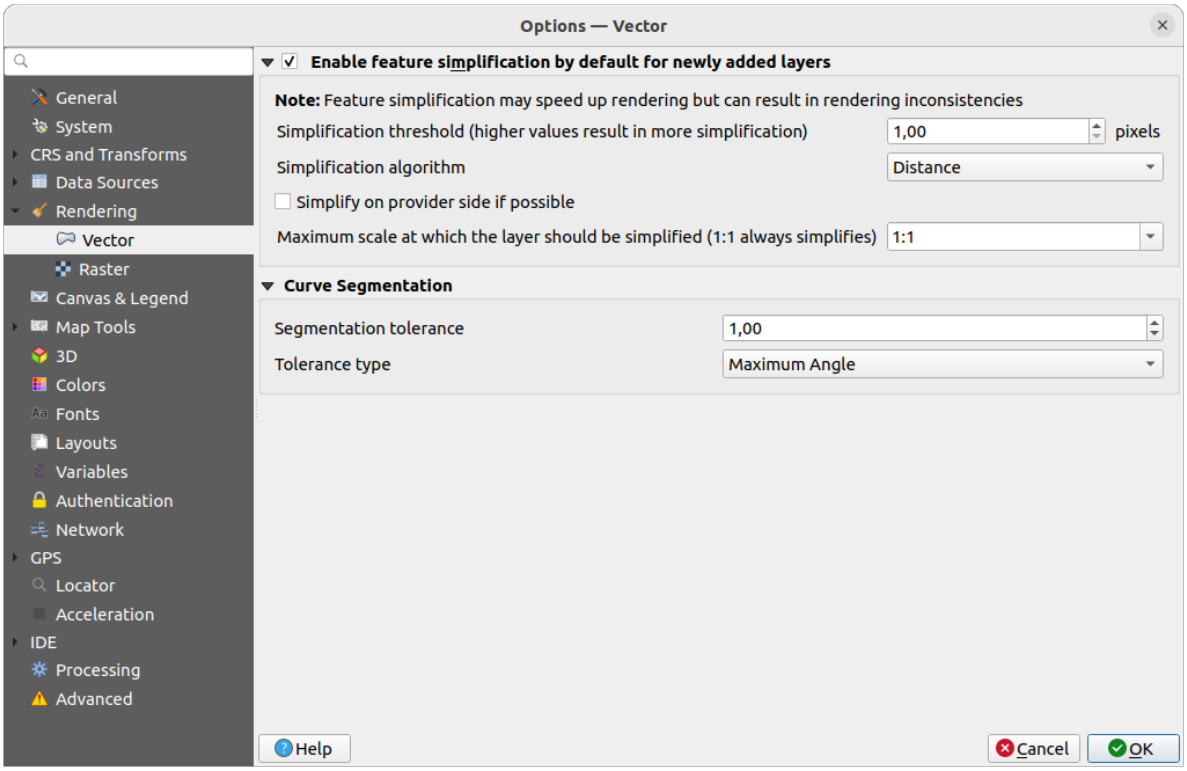


図 9.11: ベクタレンダリングの設定

- **新規追加レイヤの地物の簡素化をデフォルトで有効にする**: 地物のジオメトリを簡素化（ノード数を減らす）し、その結果、より速く表示されます。ただし、レンダリングに矛盾が生じる可能性があることに注意してください。設定可能な項目は次の通り:
  - **簡素化閾値（値が大きいほど簡素化されます）**
  - **簡素化アルゴリズム**: このオプションは、地物のローカルな簡素化を "オンザフライ" で実行し、ジオメトリのレンダリングを高速化します。この簡素化は、データプロバイダからフェッチされたジオメトリを変更することはしません。この違いは地物ジオメトリを使用する式（面積の計算など）を使用する場合に重要です。地物ジオメトリを使用する計算は、簡素化されたジオメトリではなく元のジオメトリに対して行われることが保証されます。地物の簡素化のために QGIS では「距離」（デフォルトの設定）、「グリッドにスナップ」、そして「Visvalingam」の 3 つのアルゴリズムを提供しています。
  - **可能であればプロバイダ側で簡素化する**: ジオメトリはプロバイダ（PostGIS、Oracle...）によって簡素化されます。ローカル側での簡素化と異なり、ジオメトリに関連する計算は簡素化の影響を受けます。
  - **レイヤを簡素化する最大スケール（*1:1* は常に簡素化）**

注釈: グローバル設定に加えて、特定のレイヤに対しても **レイヤのプロパティ** **レンダリング** メ

ニューから地物の簡素化の設定ができます。

- **曲線をセグメント化**
  - **セグメント化の許容差**：この設定は、円弧の描画方法を制御します。最大角度（2 つの連続した頂点と曲線の中心の間の角度を度単位で指定）または最大差（2 つの頂点のセグメントと曲線との間の距離を地図単位で指定）が **より小さい** ほど、 **より多くの直線** セグメントがレンダリング中に使用されます。
  - **許容差のタイプ** ：近似直線と曲線の間での **最大角度** または **最大差** を指定します。

ラスタレンダリングの設定

ラスタ タブにはラスタレイヤをレンダリングする特定の設定があります。

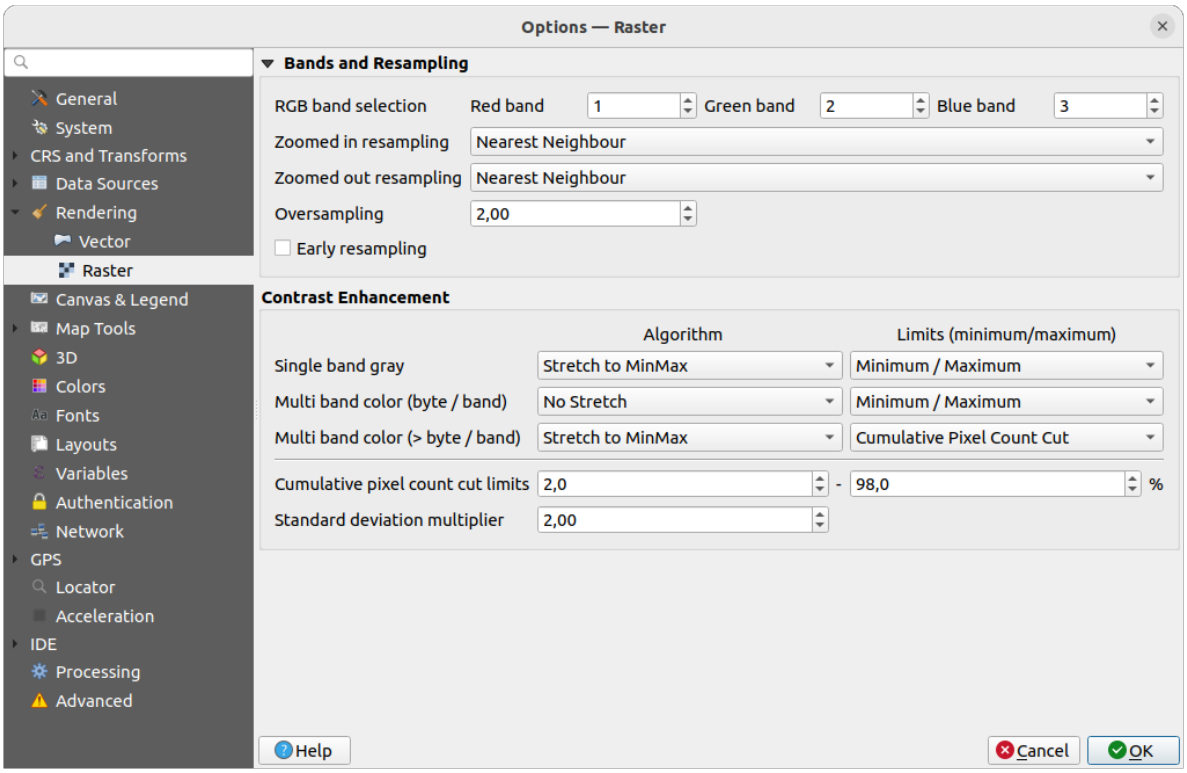


図 9.12: ラスタレンダリングの設定

バンドとリサンプリング の下には、次の設定があります:

- *RGB* **バンド選択** で、赤、緑、青のバンドの番号を定義できます。
- **拡大リサンプリング** と **縮小リサンプリング** の方法を定義できます。 **拡大リサンプリング** の場合、「最近傍（Nearest Neighbour）」、「バイリニア（Bilinear）」、「キュービック（Cubic）」の 3 つのリサンプリング方法から選択できます。 **縮小リサンプリング** の場合、「最近傍（Nearest Neighbour）」と「平均（Average）」のどちらかを選択できます。 **オーバーサンプリング** 値も設定できます（0.0 から 99.99 の間。値が大きほど QGIS の処理がより多くなります。デフォルト値は 2.0 です）。

- **早期サンプリング**: ソースの解像度がわかっているプロバイダレベルでラスタレンダリングを計算することができ、QGIS カスタムスタイルでのレンダリングでより良いズームが確保されます。これは **断面データの解釈** を使って読み込まれたタイルラスタにとても便利です。このオプションはレイヤレベルで設定することもできます（**シンボロジ** プロパティ）

コントラスト強調 オプションは、 **単バンドグレー**, **マルチバンドカラー（*1* バイト/バンド）**, **マルチバンドカラー（*1* バイト超/バンド）** に対して適用することができます。それぞれについて、次のように設定することができます:

- 使用する **アルゴリズム** の値は、「強調なし」、「最小最大範囲に引き伸ばす」、「最小最大範囲に引き伸ばしカット」、あるいは「最小最大範囲以外はカット」のいずれかです。
- 適用する **制限（最小/最大）** の値は、「累積ピクセル数でカット」、「最小 / 最大」、「平均 +/- 標準偏差」のいずれかです。

コントラスト強調 オプションには次のものもあります:

- **累積ピクセル数のカット制限**
- **標準偏差の乗数**

9.1.6 キャンバスと凡例の設定

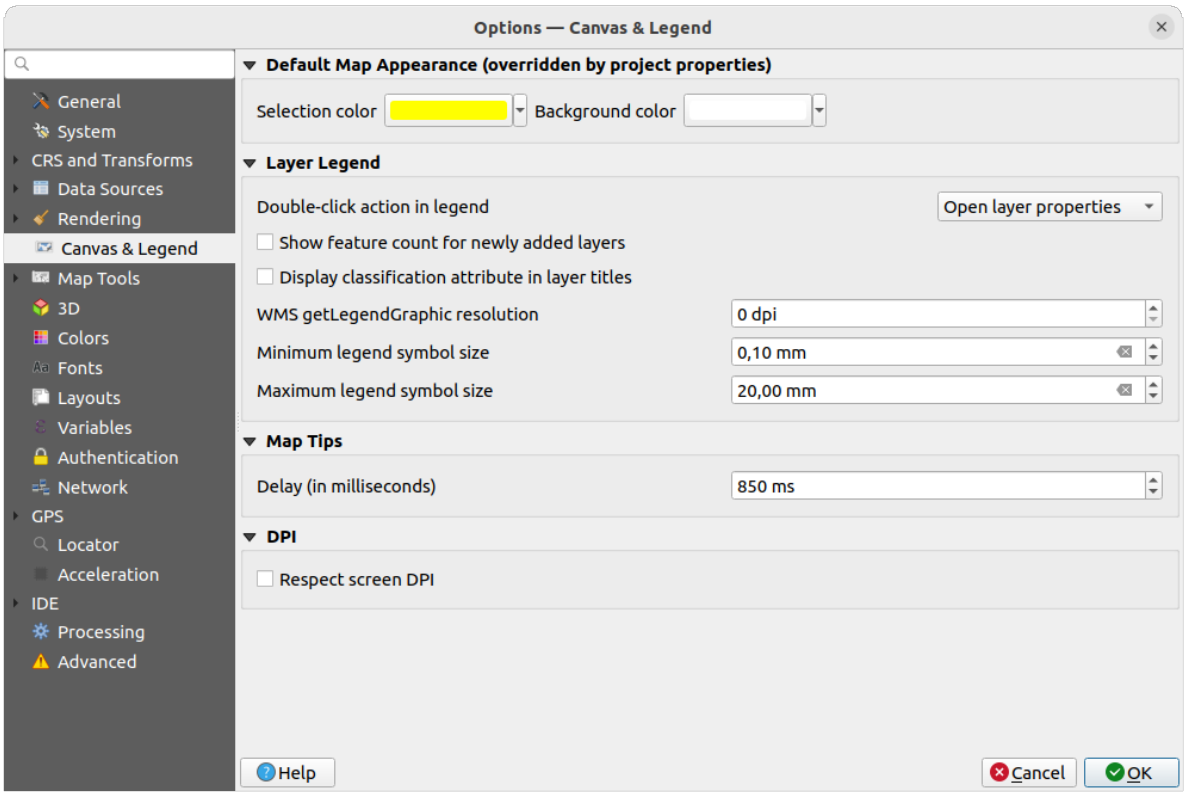


図 9.13: キャンバスと凡例の設定

このプロパティでは、以下を設定できます：

- **デフォルトの体裁（プロジェクトプロパティに上書きされます）** ： **選択物の色** および **背景色** 。
- **レイヤの凡例** のインタラクションに関する設定：
  - **凡例をダブルクリックした場合の動作**  ダブルクリックで、「レイヤのプロパティを開く」、「属性テーブルを開く」、「レイヤスタイルドックを開く」のいずれかを行うことができます。
  - **追加されたレイヤの地物数を表示**: **レイヤ** パネルに、地物の数をレイヤ名の隣に表示します。クラスの地物数がある場合は、その数も表示されます。レイヤを右クリックして、その地物数を表示/非表示することができます。
  - **レイヤタイトルに分類属性を表示** たとえばカテゴリ値による定義やルールによる定義のレンダラを適用する場合、レイヤパネルに分類に使用した属性名を表示します（詳細は **シンボロジプロパティ** を参照）。
  - *WMS getLegendGraphic* **解像度**
  - **最小凡例シンボルサイズ** と **最大凡例シンボルサイズ** は、 **レイヤ** パネル内で表示するシンボルサイズをコントロールします
- **表示までの遅延（ミリ秒）** レイヤの **地図の** *tips* が表示されるまでの遅延時間です。
- QGIS を **スクリーン** *DPI* **にする** ：これを有効にすると、QGIS はモニタの物理 DPI に応じて物理的に正確なスケールで画面上にマップキャンバスを表示しようとします。特定の表示サイズのシンボロジも同様に、正確なスケールでレンダリングされます。例えば、10mm のシンボルは、画面上で 10mm で表示されます。ただし、キャンバス上のラベルのフォントサイズは、QGIS の UI や他のアプリケーションのフォントサイズとは異なる場合があります。この設定をオフにすると QGIS は OS の論理 DPI を使用するため、システム上の他のアプリケーションと DPI が一致するようになります。ただし、キャンバスのスケールとシンボロジのサイズは画面上では物理的に不正確な場合があります。特に、高 DPI のスクリーンでは、シンボロジが非常に小さく表示される可能性があります。

  ベストな体験のためには、 **スクリーン** *DPI* **にする** を有効にすることを推奨します。特に、複数のモニタや異なる種類のモニタを使用し、視覚的に高品質なマップを表示する場合には、これを有効にしてください。 **スクリーン** *DPI* **にする** を無効にして生成される出力は、画面上での使用のみを目的とした地図に適しています。特に、フォントサイズを他のアプリケーションと一致させたい場合にはこれを無効としてください。

注釈: 印刷レイアウトのレンダリングは、 **スクリーン** *DPI* **にする** の設定の影響を受けません。印刷レイアウトは常に、ターゲット出力デバイスに指定された DPI となります。また、この設定は、OS によって報告される物理スクリーン DPI を使用することに注意してください。これは、全てのディスプレイについて正確であるとは限りません。

9.1.7 ツールの設定

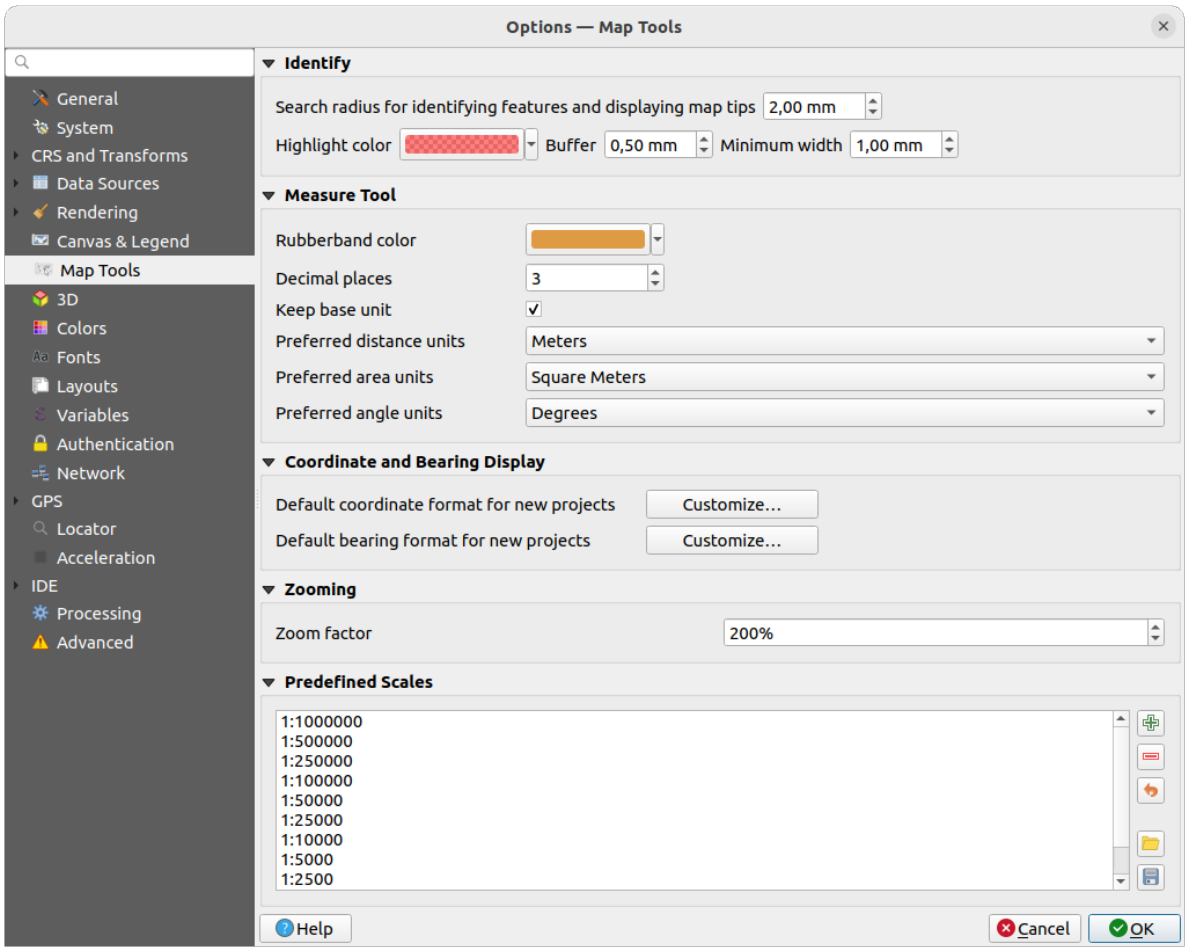


図 9.14: ツールの設定

このタブは **地物情報表示ツール** の動作に関するいくつかのオプションを提供しています。

- **地物検索・*Tips* 表示のための検索半径** は、この許容範囲内でクリックする限りにおいて、地物情報表示ツールが結果を表示する許容距離です。
- **ハイライト色** は、地物情報表示されている地物を強調表示する色を選択できます。
- **バッファ** は、地物情報表示でハイライトされている地物の輪郭に表示するバッファの距離を決定します。
- **最小幅** ハイライト表示されるオブジェクトの輪郭の厚さを決定します。

計測ツール

- 計測ツールの **ラバーバンド色** を定義します。
- **小数点以下桁数** を定義します
- **基本単位の維持** は、大きい数を自動的に変換しません（例えば、メートルをキロメートルに変換するなど）

- **優先される距離の単位**: は、「メートル」、「キロメートル」、「フィート」、「ヤード」、「マイル」、「海里」、「センチメートル」、「ミリメートル」、「度」、「地図単位」のいずれかです。
- **優先される面積の単位**: は、「平方メートル」、「平方キロメートル」、「平方フィート」、「平方ヤード」、「平方マイル」、「ヘクタール」、「エーカー」、「平方海里」、「平方センチメートル」、「平方ミリメートル」、「平方度」、「地図単位」のいずれかです。
- **優先される角度の単位**: は、「度」、「ラジアン」、「グラード」、「弧の分」、「弧の秒」、「Turns/revolutions」、「ミリラジアン（SI 定義）」、「ミル（NATO 定義）」のいずれかです。

座標と方位を表示

このセクションでは、座標と方位の表示に関する **カスタマイズ** ができます:

- **新規プロジェクトのデフォルト座標フォーマット** は、QGIS のステータスバーの **座標** ボックスに表示される数値と、 

ツールの結果の **派生した属性** セクションの座標のフォーマットです
- **新規プロジェクトのデフォルト方位フォーマット** は、マップキャンバスをパン操作した際の方向としてステータスバーに表示される値や、 

ツールに使用されるフォーマットです。

これらのオプションは、 **プロジェクトのレベル** で上書きできます。

ズーム

- ズームツールまたはマウスホイールの **ズーム倍率** を定義します。

定義済み縮尺

ここには、例えばステータスバーの **縮尺** やレイヤを表示する縮尺の設定、2 つ目の 2D マップビューの設定など、縮尺に関連したドロップダウンウィジェットでデフォルトで表示される事前定義縮尺のリストがあります。  ボタンと  ボタンを使用して、個人用の縮尺を追加・削除できます。また、縮尺を `.XML` ファイルからインポートしたり、ファイルにエクスポートすることもできます。なお、変更した内容を削除して、デフォルトのリストにリセットすることもできます。

プロジェクトのプロパティからも縮尺のリストを設定することができ、ウィジェットのグローバル設定を上書きできます。

デジタイズの設定

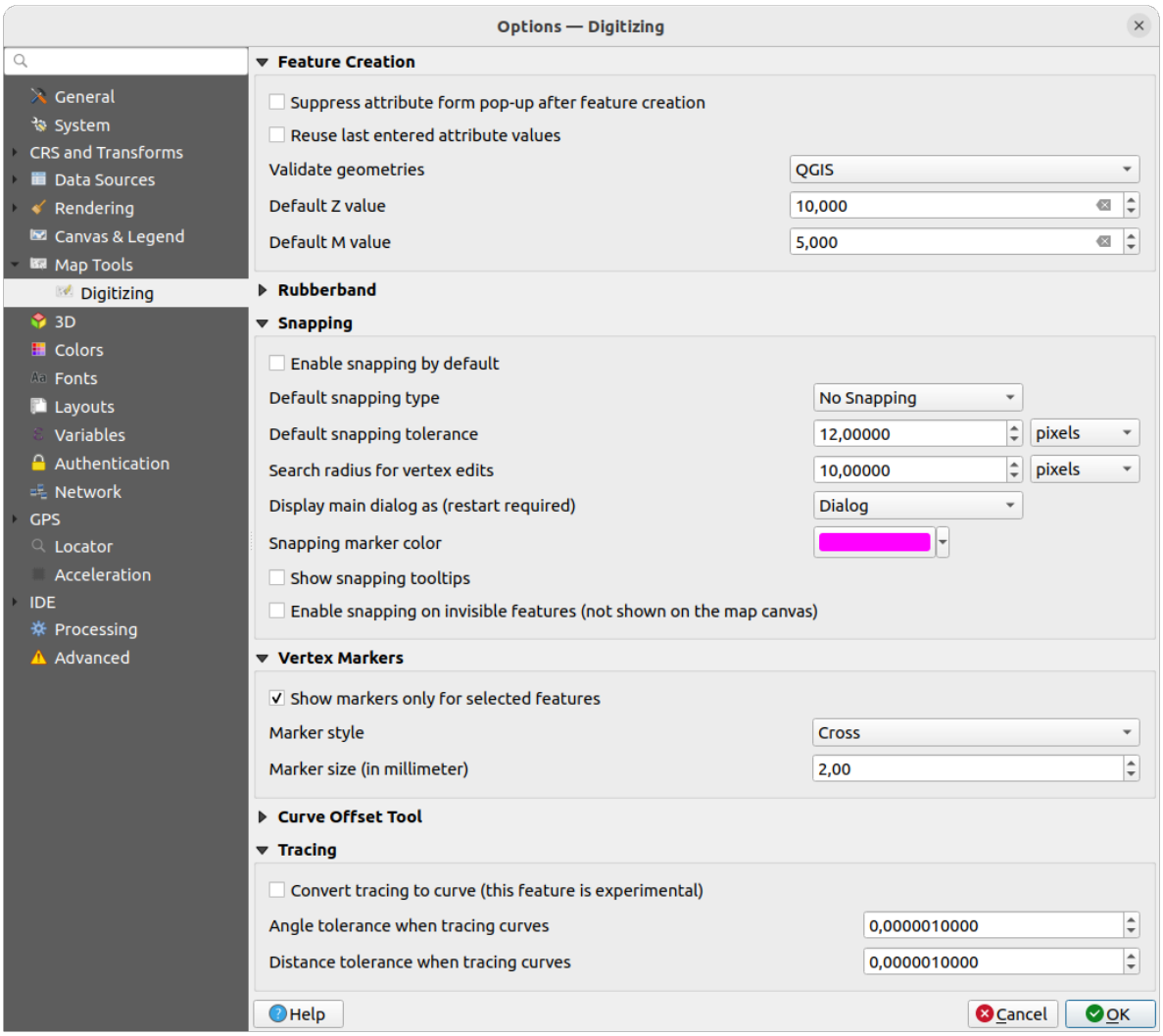


図 9.15: デジタイズの設定

このタブでは、 **ベクタレイヤを編集する** （属性とジオメトリ）際の一般的な設定を行います。

地物の作成

- **地物作成後に属性フォームをポップアップさせない** ：この選択は各レイヤのプロパティダイアログで上書きできます。
- **最後に入力した値を利用する** ：各属性の最後に使用した値を記憶し、次にデジタイズする地物のデフォルト値として使用します。これはレイヤ毎に動作します。この動作は、フィールド毎でも制御することができます（ **フィールドの動作を設定する** 参照）。
- **ジオメトリの検証** ：多数のノードを持つ複雑なラインやポリゴンを編集すると、レンダリングが非常に遅くなる可能性があります。これは、QGIS のデフォルトのジオメトリ検証は長時間かかることがあるためです。レンダリングを高速化するため、GEOS ジオメトリ検証を選択する（GEOS 3.3 以降）か、または検証をオフにすることが可能です。GEOS ジオメトリの検証ははるかに高速ですが、最初に見つかったジオメトリの問題しか報告されないという欠点があります。

選択範囲によっては、ジオメトリエラーの報告が他と異なる場合があることに注意してください（詳細は **エラーメッセージの種類と意味** を参照）

- **デフォルトの Z 値** は、新しい 3 次元地物を作成するときに使う Z 値のデフォルト値です。

ラバーバンド

- ラバーバンドの **線幅** 、 **線の色** および **塗りつぶし色** を定義します。
- **頂点の編集中にラバーバンドを更新しない**

スナップ

- **デフォルトでスナップを有効にする** プロジェクトを開いたときにスナップを有効にします
- **デフォルトのスナップ法** を定義します （「頂点」、「セグメント」、「重心点」、「セグメントの中央」、「線のエンドポイント」、「エリア」）
- **既定のスナップ許容量** を地図上の単位またはピクセルで指定します
- **頂点編集用検索半径** を地図上の単位またはピクセルで指定します
- **メインダイアログの表示（*QGIS* の再起動が必要）** : は、スナップオプションが「ダイアログ」として開かれるか、「ドック」として開かれるかを設定します。
- **スナップマーカーの色**
- **スナップツールチップを表示** は、スナップしようとしている地物のレイヤの名前などを表示します。複数の地物が重なり合っている場合に役立ちます。
- **非表示の地物に対してもスナップを有効にする** （「非表示の地物」とは、マップキャンバス上に現れていない地物のことです）

頂点マーカー

- **選択地物のみマーカーを表示**
- **マーカースタイル** 頂点マーカースタイルを定義します（「クロス」（デフォルト）、「半透明円」または「なし」）
- **マーカーの大きさ（単位ミリ）** 頂点マーカーの大きさを定義します

曲線オフセットツール

次の 3 つのオプションについては、**高度なデジタイズ** 中の  ツールを参照してください。さまざまな設定により、オフセット線の形状に影響を与えることができます。これらのオプションは GEOS 3.3 以降で使用可能です。

- **継ぎ目スタイル** :「Round」「Miter」「Bevel」
- **象限セグメント**
- *miter* **制限**

トレース

トレースを曲線に変換する（実験的） を有効にすると、デジタイズ中に曲線セグメントを作成できます。データプロバイダが曲線地物をサポートしている必要があることに留意してください。

9.1.8 3D の設定

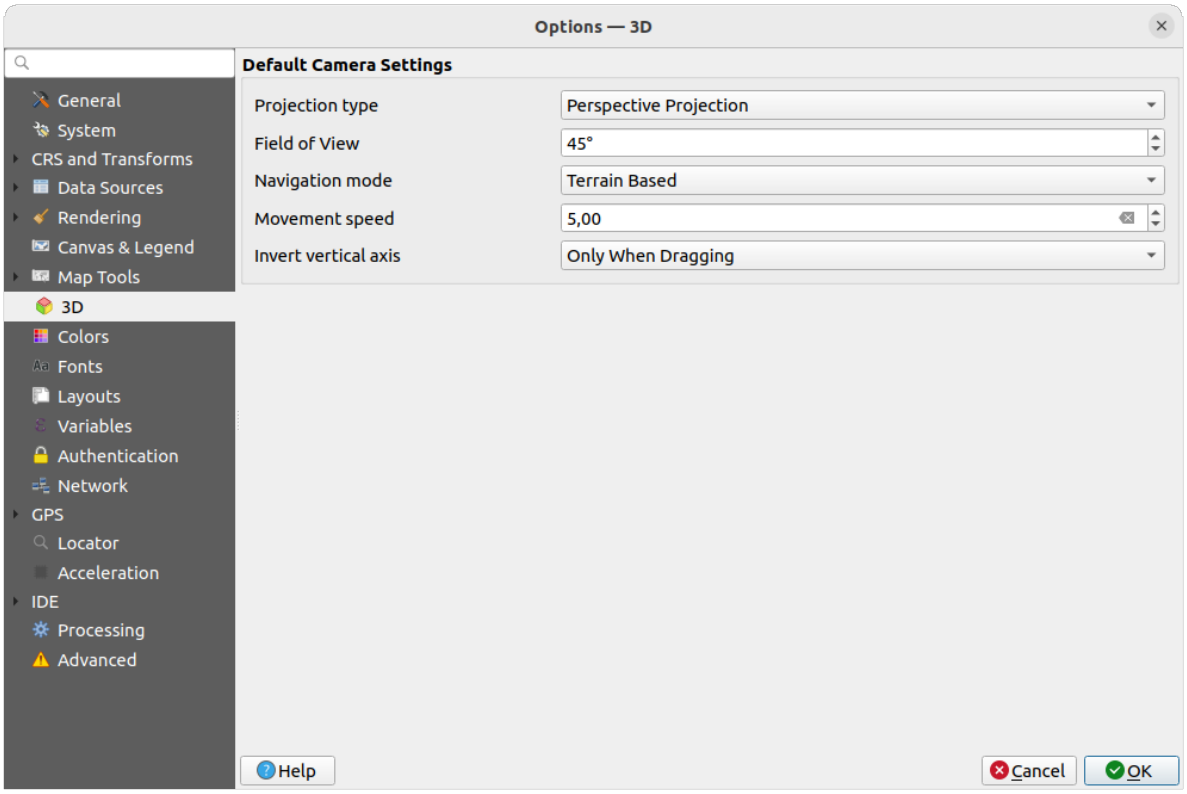


図 9.16: 3D の設定

3D メニューでは、*3D* **マップビュー** に使用される設定の一部をデフォルト設定できます。これは、**デフォルトカメラ設定** の設定を参照します：

- **投影タイプ** ：3D シーンを以下の設定で表示できます。
  - **透視投影（***Perspective***）**（デフォルト）：平行な線が遠方で合流しているように見える投影法です。物体はカメラから遠ざかるにつれて縮小して見えます。
  - **正射投影（***Orthogonal***）** ：平行な線は平行なままに見える投影法です。物体はカメラ距離にかかわらず同じ大きさに見えます。
- カメラの **視覚** ：透視投影（Perspective）モードにのみ関係し、デフォルトの垂直方向の視野を度単位で指定し、カメラが見えるシーンの範囲を決定します。デフォルトの値は 45 ° です。
- **ナビゲーションモード** ：3D シーンとやりとりするためのさまざまな手段を提供します。利用可能なモードは以下の 2 つです：
  - **地形モード** ：シーンをナビケートする際、カメラは地形の表面の固定点の周りをまわるように動きます。
  - **ウォークモード（一人称視点）**

  選択したモードによって、 **ナビゲーションコマンド** は異なります。
- **移動速度**

- **垂直軸を逆転** ：垂直軸方向の動きを、通常時の動作とは反対にします。 **ウォークモード（一人称視点）** での動きにのみ影響します。以下のいずれかに設定できます：
  - **利用しない**
  - **ドラッグ時のみ** ：クリックしてドラッグすることでカメラを回転させている時のみ、垂直軸の動きを反転させます
  - **常に** ：クリックしてドラッグでカメラを回転させる場合と、カメラの動きがカーソルにロックされている（~ キーを押す）場合の両方で、垂直軸の動きを反転させます。

9.1.9 色の設定

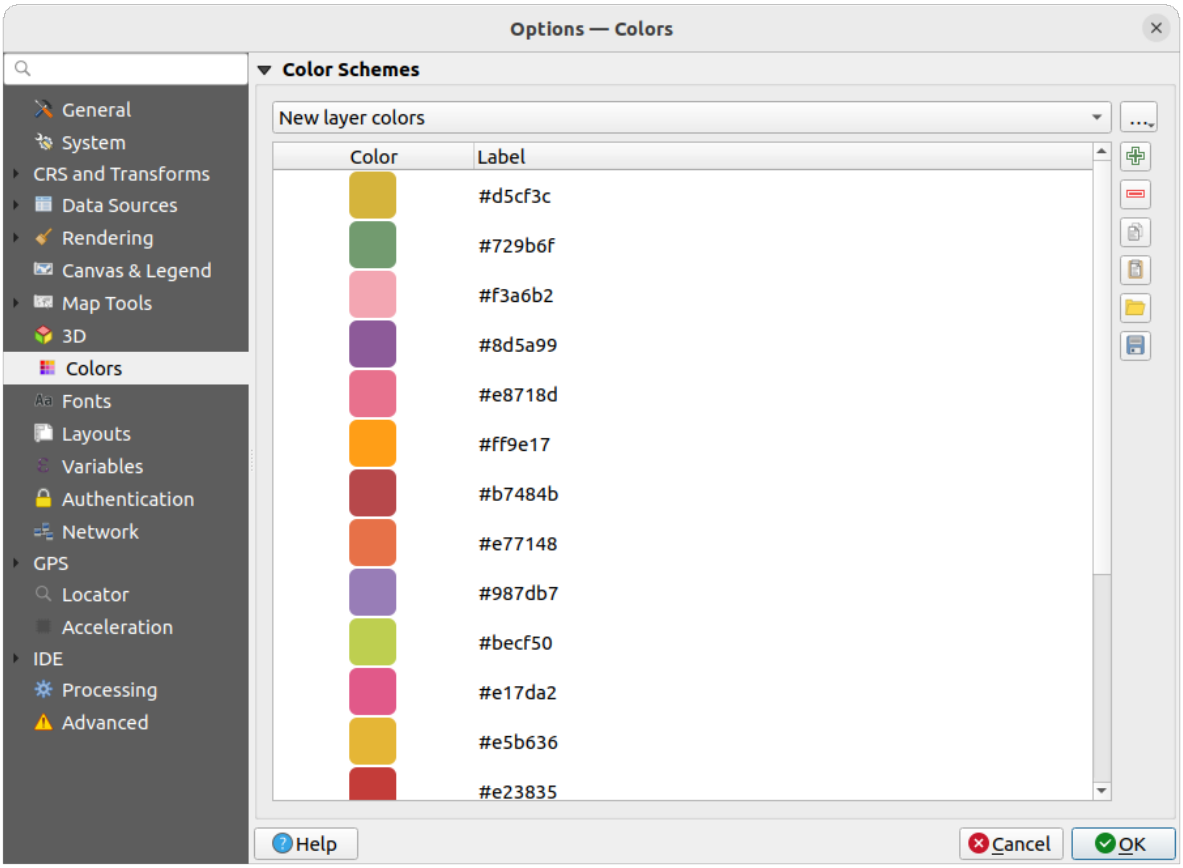


図 9.17: 色の設定

このメニューでは、アプリケーション全体で使用される色のパレットを **カラーセレクタウィジェット** で作成・更新できます。以下から選択できます。

- **最近使った色** ：最近使用した色を表示します
- **標準色** ：デフォルトの色パレットです
- **プロジェクトの色** ：現在のプロジェクトに固有の色のセットです（詳細は **スタイルプロパティ** を参照してください）。

- *New layer colors*：新しいレイヤが QGIS に追加されたときにデフォルトで使用される色のセットです。
- あるいは、パレットコンボボックスの横にある ... ボタンを押して、カスタムパレットを新規作成もしくはインポートできます。

デフォルトでは、 **最近使った色** 、 **標準色**、および **プロジェクトの色** パレットは削除できず、色ボタンドロップダウンに表示されるように設定されています。カスタムパレットは **色ボタンに表示** オプションで、ウィジェットに追加できます。

どのパレットでも、フレームの横にあるツールセットを使って色のリストを管理することができます。

- **色を追加** あるいは **色を削除**
- **色のコピー** あるいは **色を貼り付け**
- **ファイルから色のインポート** あるいは **色のエクスポート** で色のセットを `.gpl` ファイルからインポート / ファイルへエクスポート

リスト内の色をダブルクリックして **カラーセレクタ** ダイアログで色を調整したり、置き換えたりできます。 **ラベル** 列をダブルクリックして、色の名前を変更することもできます。

9.1.10 フォントの設定

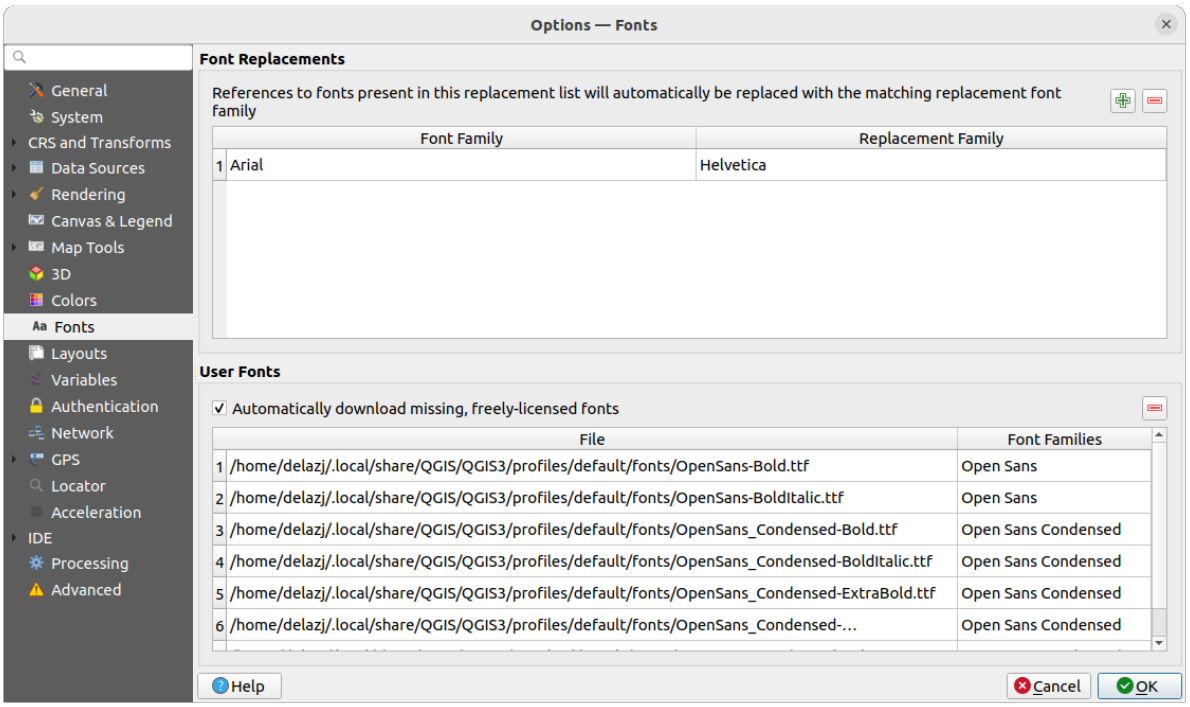


図 9.18: フォントの設定

フォント タブでは、プロジェクト全体で使用するフォントの管理をサポートします:

- **フォント置換**: プロジェクトやスタイルを読み込んだ際に適用する自動的なフォント置換に関するリストの入力ができます。フォント置換を使用することで、異なるオペレーティングシステム間でプロジェクトやスタイルをより良くサポートすることができます（例えば "Arial" を "Helvetica" に置換）。

- **ユーザーフォント**: **ユーザープロファイル** フォルダの下の `fonts` フォルダには TTF フォントや OTF フォントを配置することができます。これらのフォントは QGIS の起動時に自動的に読み込まれます。社内環境ではオペレーティングシステムのレベルでのフォントのインストールはしばしばブロックされてしまいますが、これにより、オペレーティングシステムのレベルでインストールすることなく QGIS でフォントを使用することができます。このパネルには配置したすべてのユーザーフォントが一覧表示され、これまでに配置したユーザーフォントを管理（つまり削除）することができます。

  フリーフォントを自動ダウンロード することができます。例えば、プロジェクトやスタイルを開いたり、現在利用できないフォントを参照するベクタタイルレイヤを読み込もうとすると、URL 経由でダウンロードできるフリーライセンスフォントのハードコード化されたリストを参照し、ユーザープロファイルのフォントディレクトリにフォントを（フォントライセンスの通知付きで）自動的にダウンロードできるか判断します。

9.1.11 レイアウトの設定

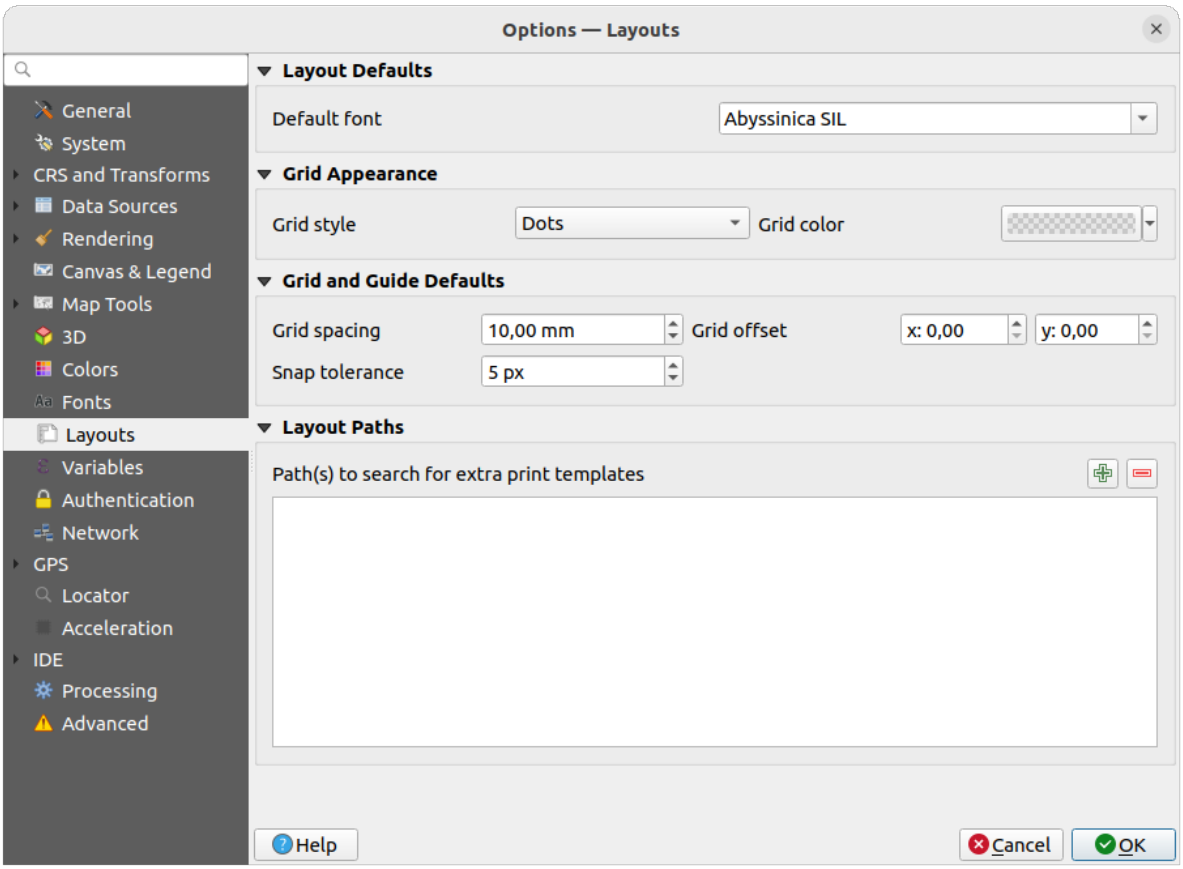


図 9.19: レイアウトの設定

レイアウトのデフォルト

印刷レイアウト で使用される **デフォルトフォント** を定義できます。

グリッドの外観

- **グリッドスタイル**（'塗りつぶし'、'点'、'十字'）を定義します

- **グリッド色** を定義します

グリッドとガイドのデフォルト

- **グリッド間隔** を定義します
- X と Y の **グリッドオフセット** を定義します
- **スナップ許容量** を定義します

レイアウトのパス

- **追加のプリントテンプレートを探すパス** の定義：新しい印刷レイアウトを作成する際に使用する、カスタムの印刷レイアウトテンプレートが保存されたフォルダのリストです。

9.1.12 変数の設定

変数 タブには、グローバルレベルで利用可能なすべての変数がリストされています。

ここではグローバルレベルの変数の管理もできます。 ボタンをクリックして、グローバルレベルの新しいカスタム変数を追加します。同様に、リストからカスタムグローバルレベル変数を選択し、 ボタンを押して削除します。

変数の詳細については、 **値を変数に格納する** セクションを参照してください。

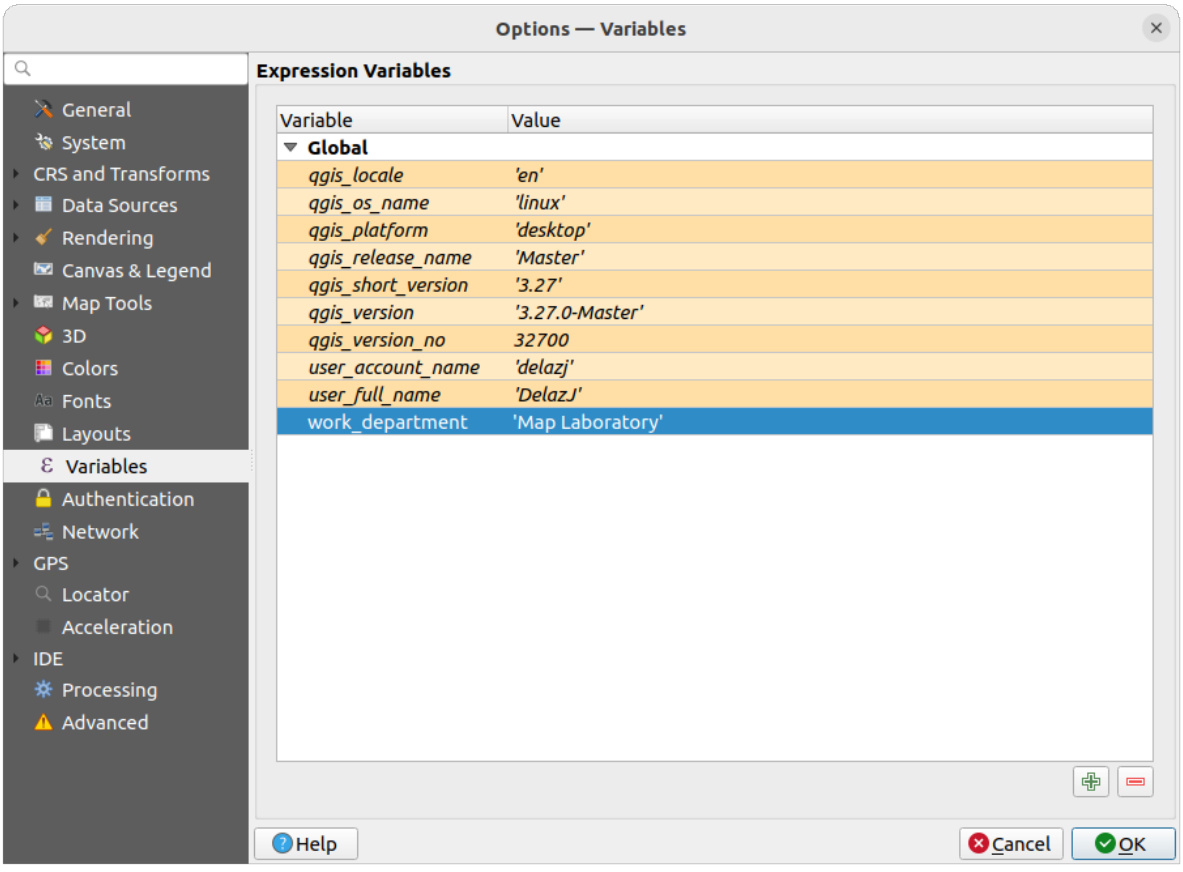


図 9.20: 変数の設定

9.1.13 認証の設定

認証 タブでは、認証設定を設定し、PKI 証明書を管理できます。詳細は **認証システム** を参照してください。

認証情報を管理するために、フレームの横にある以下のツールを使用できます:

- 認証構成を追加
- 選択した認証構成を削除
- 選択した認証構成を編集

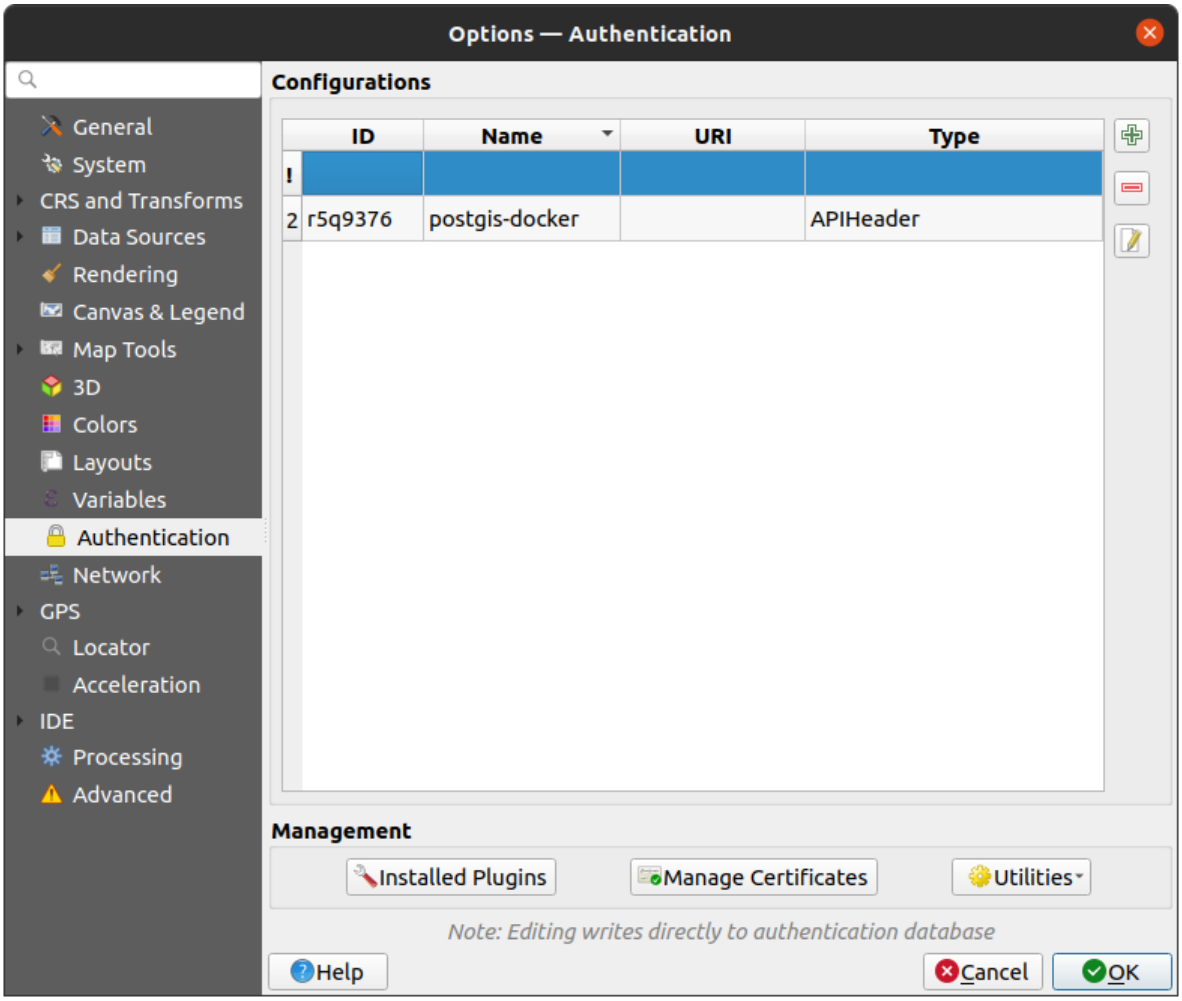


図 9.21: 認証の設定

9.1.14 ネットワークの設定

一般情報

- **ネットワークリクエストのタイムアウト（ミリ秒）** を定義します。デフォルトは 60000 ミリ秒です
- *WMS Capabilities* **のデフォルト有効期間（***hours***）** を定義します。デフォルトは 24 時間です
- *WMS-C/WMTS* **タイルのデフォルト有効期間（***hours***）** を定義します。デフォルトは 24 時間です
- **タイル / 地物リクエストエラーの場合の最大再試行** の回数を指定します
- *User-Agent* を定義します

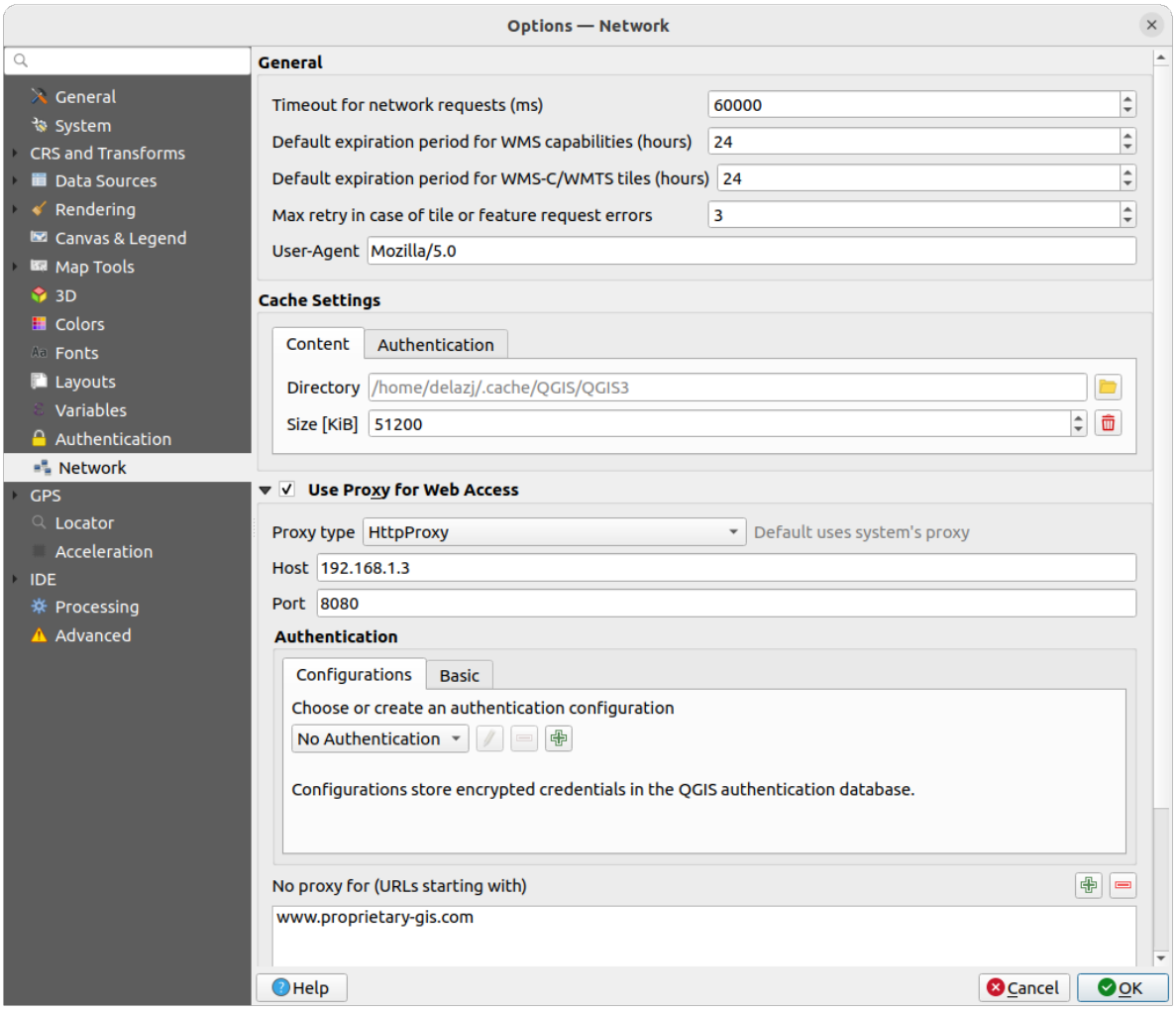


図 9.22: ネットワークとプロキシの設定

キャッシュ設定

キャッシュの **ディレクトリ** と **サイズ** *[KiB]* を定義します。また、 *SSL* **エラーで接続認証キャッシュを自動的に消去する（推奨）** ためのツールを提供します。

ウェブ接続のプロキシ

- **ウェブ接続にプロキシを使用**

- 必要に応じて **プロキシタイプ**  を設定し、「ホスト」と「ポート番号」を定義します。利用可能なプロキシタイプは以下の通りです。
  - *Default Proxy*: プロキシはシステムのプロキシ設定をもとに決定されます
  - *Socks5Proxy*: あらゆる種類の接続に対応した一般的なプロキシです。TCP、UDP、ポートへのバインディング（入力コネクション）と認証をサポートします。
  - *HttpProxy*: "CONNECT" コマンドで実装され、外向きの TCP コネクションのみサポートしています。また、認証をサポートしています。
  - *HttpCachingProxy*: 通常の HTTP コマンドを使って実装され、HTTP リクエストのコンテキストでのみ役に立ちます。
  - *FtpCachingProxy*：FTP プロキシを使用して実装されています。FTP リクエストのコンテキストでのみ役に立ちます。

プロキシの資格情報は **認証ウィジェット** を使用して設定します。

プロキシ設定の下のテキストボックスにいくつかの例外 URL を追加することができます（ 図 9.22 を参照)。ターゲット URL がこのテキストボックスにリストされた文字列の 1 つで始まる場合、プロキシは使用されません。

さまざまなプロキシ設定に関するより詳細な情報が必要な場合は、 https://doc.qt.io/archives/qt-5.9/qnetworkproxy.html#ProxyType-enum にある Qt ライブラリのマニュアルを参照してください。

Tip: プロキシの利用

プロキシを使うのは時に厄介なことがあります。上記のプロキシタイプを使って「試行錯誤」し、自分のケースでうまくいくかどうかを確認ながら進めていくのがよいでしょう。

9.1.15 GPS の設定

GPS 可視化オプション

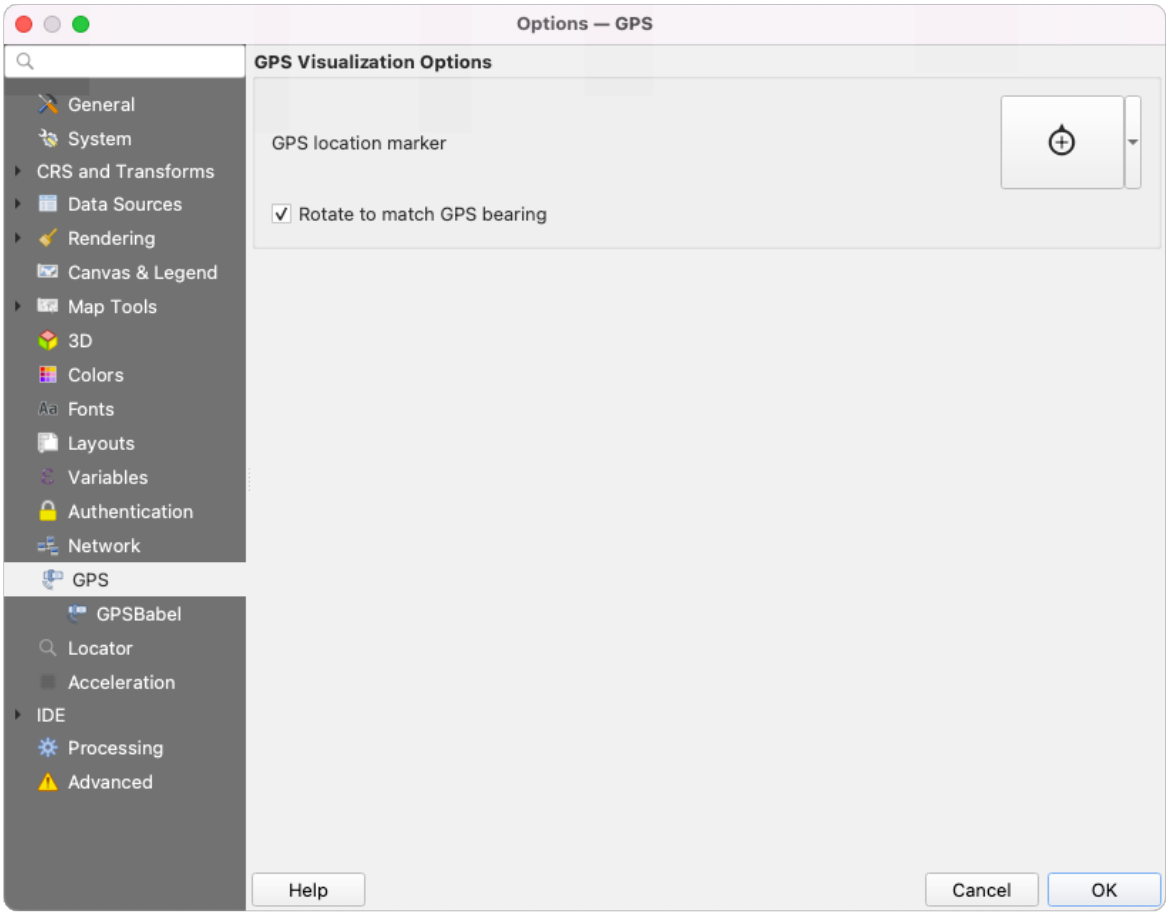


図 9.23: GPS の設定

このダイアログでは、 *QGIS* **に** *GPS* **機器を接続する** 際の表示を設定できます。

- *GPS* **位置マーカー** は、現在の GPS 位置の表示に使用するマーカーシンボルを設定します。
- *GPS* **の方角に合わせて回転** は、マーカーシンボルを GPS の方角に合わせて回転させるかを設定します。

GPSBabel

GPSBabel は、Garmin や Magellan といった一般的な GPS 受信機と、Google Earth や Basecamp といったマッピングプログラムの間でウェイポイント、トラック、ルートを変換します。文字通り、何百もの GPS 受信機とプログラムをサポートしています。QGIS は、これらのデバイスとのやり取りや **データの操作** を GPSBabel に頼っています。

1. 最初に、 *GPSBabel* **のパス** で、GPSBabel のバイナリへのパスを定義する必要があります。
2. 次に、デバイスを追加します。デバイスのリストは 新規デバイスを追加 ボタンや デバイスを削除 ボタンを使用して更新できます。

3. 各デバイスについて、以下の設定ができます:

   - **デバイス名** の設定
   - QGIS がデバイスとのやり取りに使用するさまざまな **コマンド** の設定。例えば:
     - デバイスから **ウェイポイントをダウンロード**
     - デバイスへ **ウェイポイントをアップロード**
     - デバイスから **ルートをダウンロード**
     - デバイスへ **ルートをアップロード**
     - デバイスから **トラックをダウンロード**
     - デバイスへ **トラックをアップロード**

     コマンドは通常、GPSBabel のコマンドですが、GPX ファイルを作成することのできるその他のコマンドラインプログラムを使用することもできます。QGIS はコマンドを実行する際に `%type` 、 `%in` 、 `%out` のキーワードを置き換えます。

     例えば、ダウンロードコマンド `gpsbabel %type -i garmin -o gpx %in %out` でデバイスタイプを作成し、それを使用してポート `/dev/ttyS0` からファイル `output.gpx` にウェイポイントをダウンロードすると、QGIS はキーワードを置き換えて、コマンド `gpsbabel -w -i garmin -o gpx /dev/ttyS0 output.gpx` を実行します。

     自身のユースケースに特有のコマンドラインオプションについては、GPSBabel のマニュアルを確認してください。

新しいデバイスタイプを作成することができたら、GPS のダウンロード・アップロードアルゴリズムのデバイスリストに表示されます。

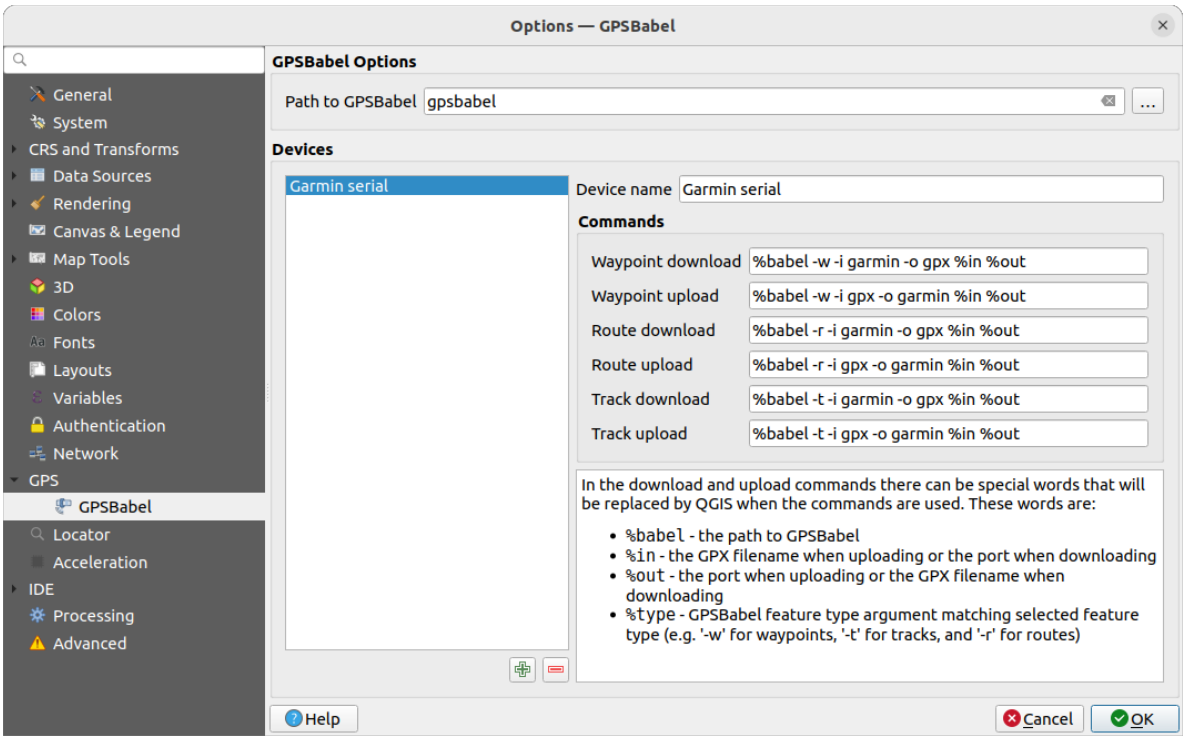


図 9.24: GPS Babel の設定

9.1.16 ロケータの設定

ロケータ タブでは、**ロケータバー** の設定を行えます。 これはステータスバーにあるクイック検索ウィジェットで、アプリケーション内の検索を実行できます。デフォルトのフィルタ（や接頭辞）なども設定できます。

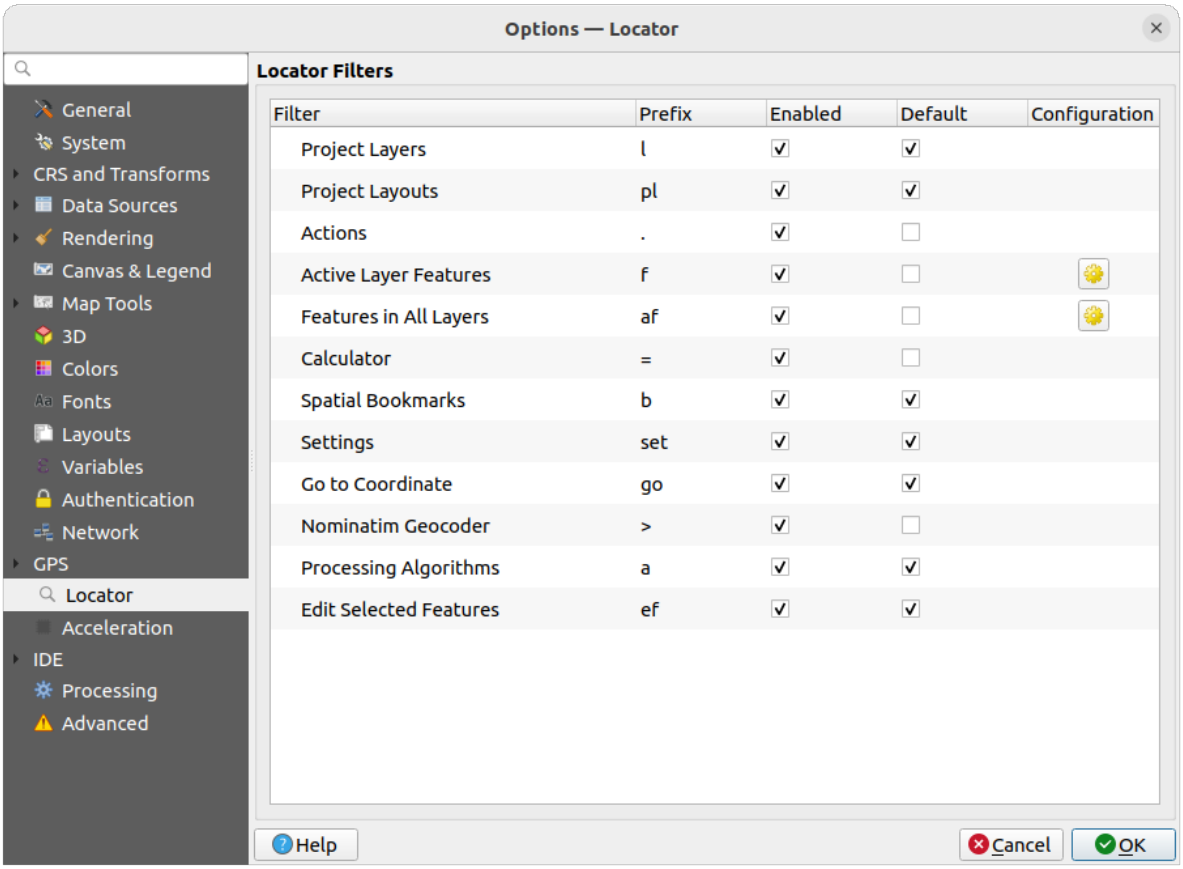


図 9.25: ロケータの設定

- **プロジェクトのレイヤ** (`l`): **レイヤ** パネル内のレイヤを検索し、選択します。
- **プロジェクトのレイアウト** (`pl`): 印刷レイアウトを検索し、開きます。
- アクション (`.`): QGIS のアクションを検索し、実行します。アクションには、QGIS のツールやメニュー、パネルを開く等があります。
- **アクティブレイヤの地物** (`f`): 現在アクティブなレイヤから任意のフィールドで一致する属性を検索し、選択した地物にズームします。 を押すと、結果の最大数を設定できます。
- **全てのレイヤの地物** (`af`): 各 **検索可能なレイヤ** の **表示名** でマッチする属性を検索し、選択した地物にズームします。 を押すと、結果の最大数とレイヤごとの結果の最大数を設定できます。
- **計算機** (=): 任意の QGIS 式の評価ができ、有効な式であれば、結果をクリップボードにコピーするオプションが表示されます。
- **空間ブックマーク** (b): ブックマーク範囲を検索し、ズームします。
- **設定** (`set`): プロジェクトやアプリケーション全体のプロパティダイアログを参照し開きます。
- **座標へ** (go): コンマまたは空白区切りの xy 座標の組で定義される場所、あるいはフォーマットされた URL（例：OpenStreetMap、Leaflet、OpenLayer、Google Maps、...）で定義される場所へマップキャンバスを移動します。 座標値は WGS 84（ `epsg:4326` ）またはマップキャンバスの CRS です。
- *Nominatim Geocoder* (>): OpenStreetMap Foundation の Nominatim ジオコーディングサービスを使用したジオコーディングを行います。

- プロセシングアルゴリズム (a): プロセシングアルゴリズムダイアログを検索し開きます。
- **選択地物を編集する** (ef): プロセシングアルゴリズムへのクイックアクセスを提供し、互換性のある *in-place* **編集** のプロセシングアルゴリズムをアクティブレイヤで実行します。

このダイアログでは、以下の設定ができます：

- フィルタの **接頭辞** すなわちフィルタ利用のトリガーとなるキーワードをカスタマイズします
- フィルタの **有効化** の設定：フィルタを検索で使用し、ロケータバーのメニューでショートカットを有効とするかどうかを設定します
- フィルタの **デフォルト** の設定： フィルタを使用しない検索では、デフォルトフィルタのカテゴリによる結果のみが返されます
- 一部のフィルタには、検索結果の数を設定する方法があります。

デフォルトのロケータフィルタのセットはプラグインによって拡張できます。例えば、OSM nominatim 検索や直接データベース検索、レイヤカタログ検索などがあります。

9.1.17 高速化の設定

OpenCL による高速化の設定です。

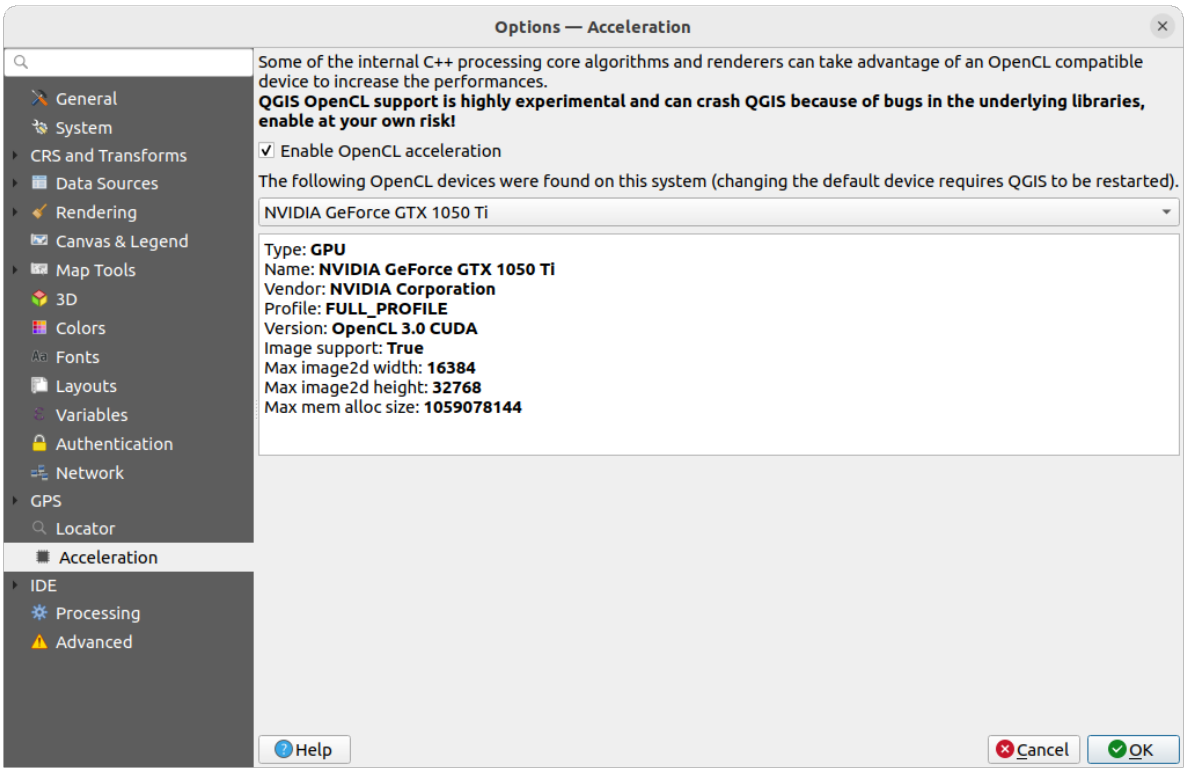


図 9.26: 高速化の設定

お使いのハードウェアやソフトウェア環境にもよりますが、OpenCL アクセラレーションを有効にするためには、追加のライブラリをインストールする必要がある場合があります。

9.1.18 IDE の設定

コードエディタの設定

コードエディタ タブでは、コードエディタウィジェット（Pyhon 対話型コンソール、コードエディタ、式ウィジェット、関数エディタ等）の見た目や動作を制御することができます。

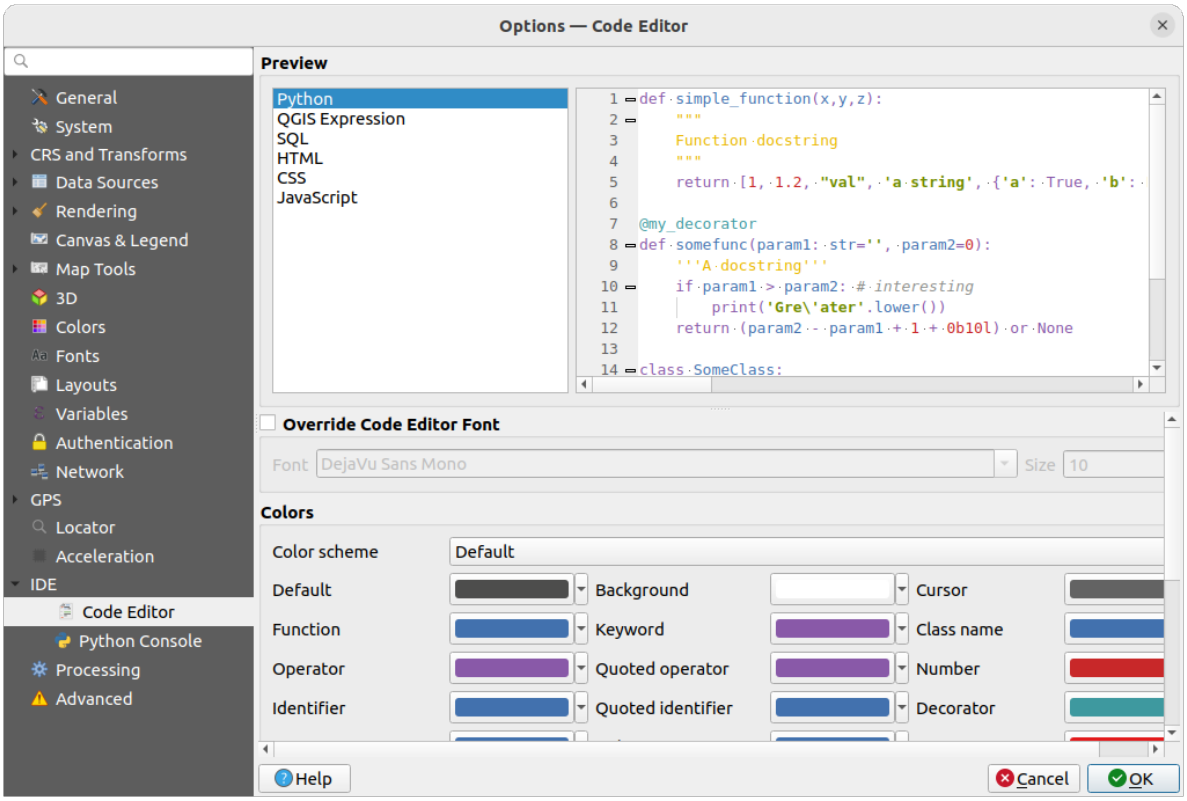


図 9.27: コードエディタの設定

ダイアログの上部にあるウィジェットでは、さまざまなプログラミング言語（Python、QGIS 式、HTML、SQL、JavaScript）で現在の設定をライブプレビューできます。設定を調整するのに便利です。

- **コードエディタのフォント上書き** にチェックを入れると、デフォルトの **フォント** ファミリーと **大きさ** を修正できます。
- **色** グループでは、以下の設定ができます：
  - **配色** の選択：定義済みの設定は デフォルト 、 低コントラスト（暗） 、 低コントラスト（明） です。色を変更すると カスタム スキームとなり、定義済みのスキームを選択することでリセットできます。
  - コメント、クォート、関数、背景など、コードの各要素の **色** を変更できます。

Python コンソールの設定

Python **コンソール** 設定では、Python エディタ（**対話型コンソール**、**コードエディタ**、**プロジェクトのマクロ**、**カスタム式** 等）の動作の管理と制御を行えます。このメニューには、下記の オプション... ボタンからもアクセスできます。

- *Python* **コンソール** ツールバー
- *Python* **コンソール** ウィジェットのコンテキストメニュー
- コードエディタのコンテキストメニュー

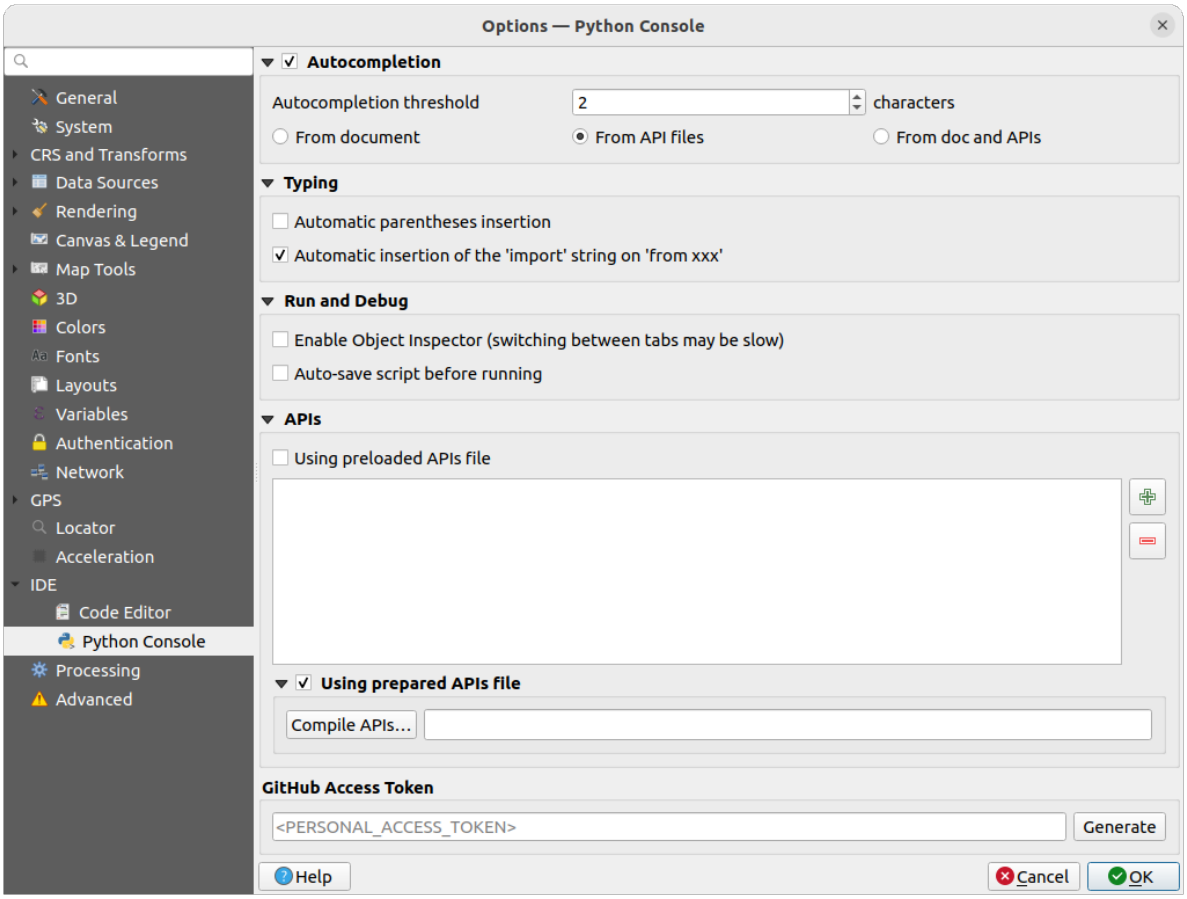


図 9.28: Python コンソールの設定

以下の設定ができます：

- **入力補完（オートコンプリート）**：コードの入力補完を有効にします。現在のドキュメント、インストールされた API ファイル、またはその両方から自動補完されます。
  - **入力補完の閾値**：自動補完リストを表示する閾値を（文字数で）設定します。
- **タイピング**
  - **カッコの自動挿入**：左括弧を入力した時の右括弧の自動入力を有効にします
  - *'from xxx'* **に** *'import'* **文字列を自動挿入する**：インポート指定の際に 'import' 文字列の挿入を有効にします。

- 実行とデバッグ
  - **オブジェクトインスペクタを有効にする（タブの切り替えが遅くなる場合があります）**
  - **実行前にスクリプトを自動保存する** ：スクリプトの実行時に自動的にスクリプトを保存します。このアクションは一時ファイルを（システムディレクトリの一時領域に）保存し、実行後には自動的に削除されます。

API では、以下の設定ができます：

- **プリロードされた *API* ファイルを使用** ：プリロードされた API ファイルを使用するかどうかを選択できます。 これがチェックされていない場合には API ファイルを追加することができ、また、事前に準備された API ファイルを使用するかどうかを選択することもできます（次のオプションを参照）。
- **事前に準備された *API* ファイルを利用する** ：チェックした場合、選択した `*.pap` ファイルがコード補完に使用されます。準備された API ファイルを生成するには、少なくとも 1 つの `*.api` ファイルをロードし、 ***API* をコンパイル**... ボタンをクリックしてそれをコンパイルする必要があります。

GitHub **アクセストークン** では、Python コードエディタ内からコードスニペットを共有するための個人アクセストークンを生成できます。詳細については、 GitHub authentication を参照してください。

9.1.19 プロセシングの設定

プロセシング タブでは、QGIS プロセシングフレームワークで使用されるツールやデータプロバイダの一般的な設定を行うことができます。詳細は、 *QGIS* **プロセシングフレームワーク** を参照してください。

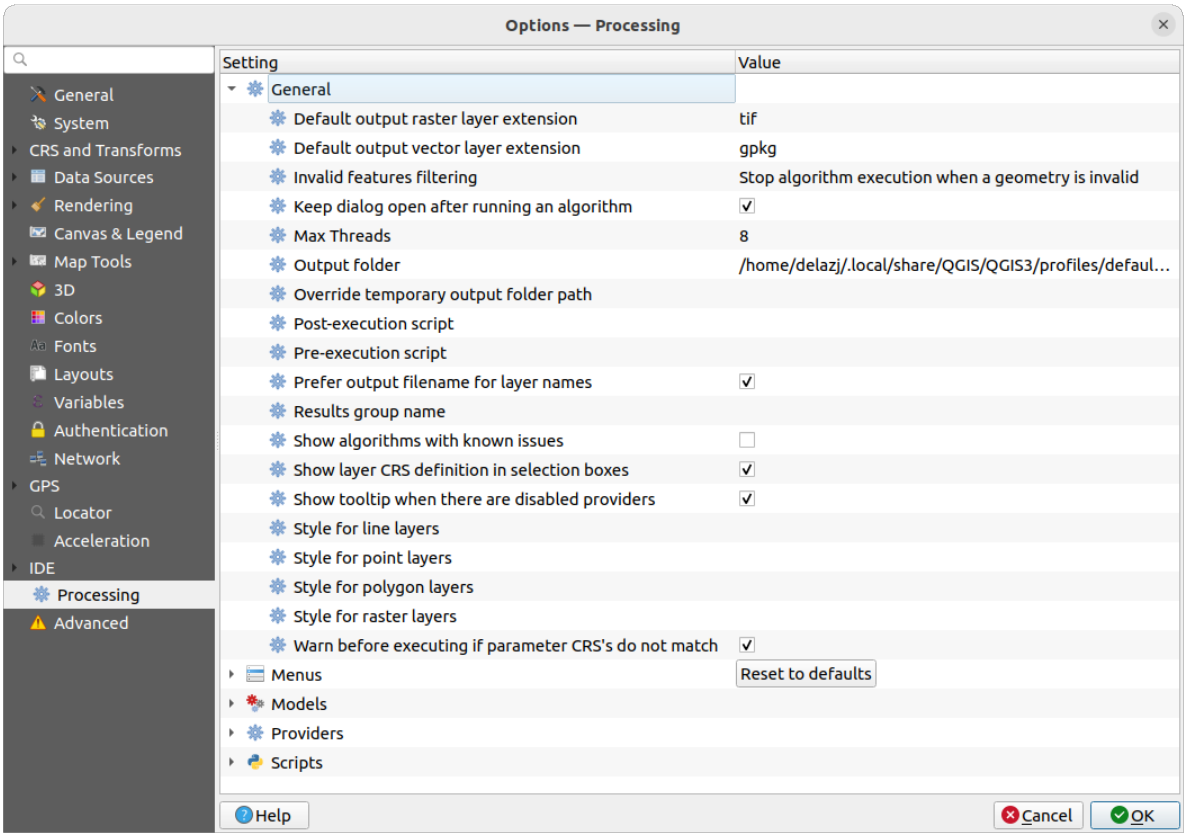


図 9.29: プロセシングの設定

9.1.20 詳細設定

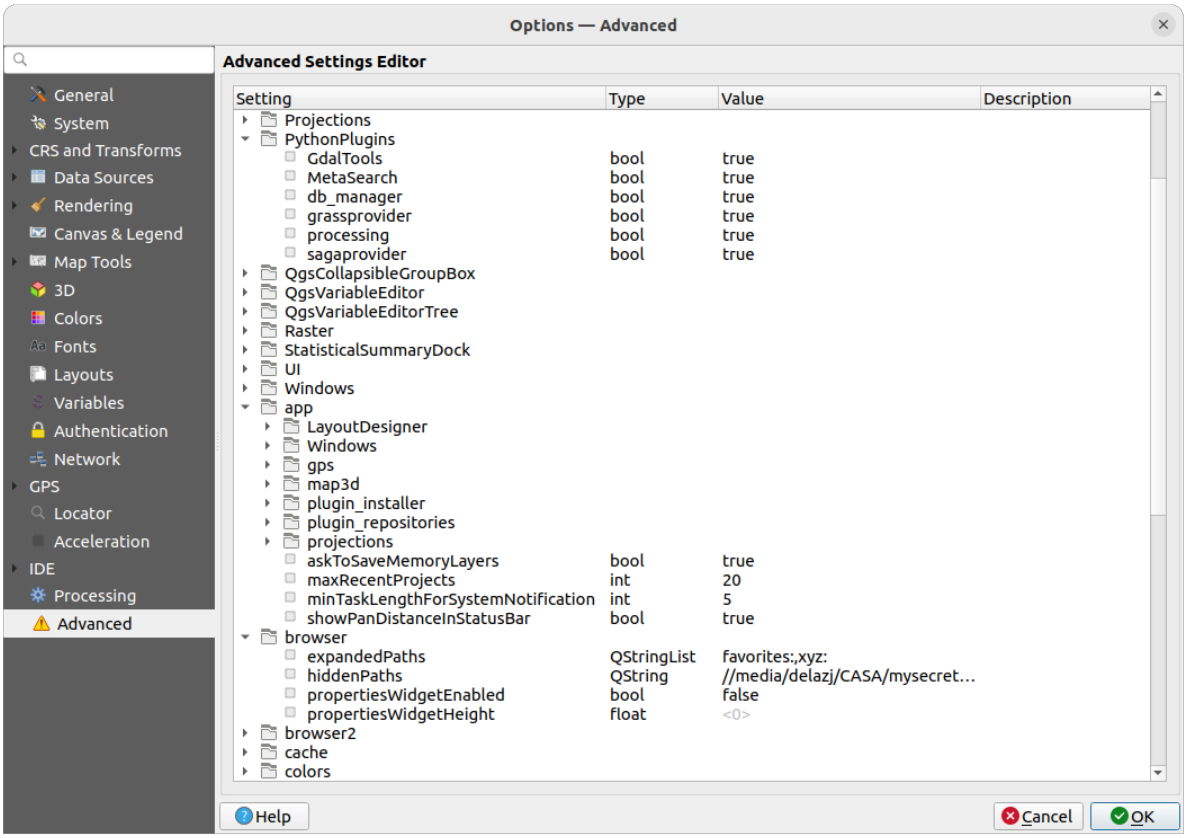


図 9.30: 詳細設定

QGIS に関連するすべての設定（UI、ツール、データプロバイダ、プロセシングアルゴリズムの構成、デフォルト値やパス、プラグインのオプション、式、ジオメトリチェックなど）は、アクティブな **ユーザープロファイル** ディレクトリの `QGIS/QGIS3.ini` ファイルに保存されます。このファイルを他のインストール環境にコピーすることで、設定を共有することができます。

QGIS 内では、 **詳細設定** タブの **詳細設定エディタ** を使用してこれらの設定を管理する方法を提供しています。慎重に設定変更することを約束したら、ウィジェットには既存のすべての設定のツリーが表示され、その値を編集することができるようになります。設定やグループの上で右クリックすると、設定やグループを削除することができます（設定やグループを追加するには、 `QGIS3.ini` ファイルを編集する必要があります）。変更は自動的に `QGIS3.ini` ファイルに保存されます。

警告: 「高度な設定」タブの設定をやみくもに使用しないこと

変更は自動的に適用されるため、このダイアログで項目を変更する際には注意が必要です。知識のないまま変更を行うと、さまざまな形で QGIS のインストール環境が壊れる可能性があります。

9.2 ユーザープロファイルの操作

設定 ▸ **ユーザープロファイル** メニューは、ユーザープロファイルを設定したり、ユーザープロファイルにアクセスする機能を提供します。ユーザープロファイルとは、単一のフォルダに保存される一元管理されたアプリケーション構成で、以下のものを含みます。

- ロケール、投影法、認証設定、カラーパレット、ショートカットなどを含むすべての **グローバル設定**
- GUI の構成と **カスタマイズ**
- 測地系変換のためにインストールされたグリッドファイルやその他の proj ヘルパーファイル
- インストールされた **プラグイン** とその構成
- プロジェクトのテンプレートと、保存されたプロジェクトの履歴およびその画像プレビュー
- **プロセシングの設定** やログ、スクリプト、モデル

デフォルトでは、QGIS インストールには `default` という名前のユーザープロファイルが一つだけ含まれています。ただし、ユーザープロファイルは以下の方法で必要な数だけ作成できます：

1. **新規プロファイル**... エントリをクリックします。
2. プロファイル名を指定するように求められ、同じ名前のフォルダが `~/<UserProfiles>/` の下に作成されます。
   - `~` は **HOME** ディレクトリを表します。 Windows では、通常は `C:\Users\<username>` のような名前のフォルダです。
   - `<UserProfiles>` は、メインプロファイルフォルダを表します。
     - `.local/share/QGIS/QGIS3/profiles/`
     - `%AppData%\Roaming\QGIS\QGIS3\profiles\`
     - `Library/Application Support/QGIS/QGIS3/profiles/`

   ユーザープロファイルフォルダは、 **アクティブなプロファイルフォルダを開く** を使用して QGIS 内から開くことができます。
3. クリーンな構成を使用して、QGIS の新しいインスタンスが起動されます。その後、カスタム構成を設定することができます。

QGIS インストール環境に複数のプロファイルがある場合、アプリケーションのタイトルバーにアクティブなプロファイル名が角括弧囲みで表示されます。

各ユーザープロファイルには個別の設定、プラグイン、および履歴が含まれているため、異なるワークフローで使い分けたり、デモ用途や、同じマシンでユーザーが複数いる場合、あるいは設定のテスト用などに最適です。 **設定** ▸ **ユーザープロファイル** メニューで選択することで、あるプロファイルから別のプロファイルへと切り替えができます。 **コマンドライン** から特定のユーザープロファイルで QGIS を実行することもできます。

最後に閉じられた QGIS セッションのプロファイルは、変更されない限りは次の QGIS セッションで使用されます。

Tip: 新しいユーザープロファイルで QGIS を実行して、バグの持続性を確認する

QGIS の一部の機能で奇妙な動作が発生した場合には、新しいユーザープロファイルを作成してコマンドを再度実行してください。バグは、現在のユーザープロファイルに残っているものが関係していることがあります。新しいユーザープロファイルを作成すると、新しい（クリーンな）プロファイルで QGIS を再起動するため、バグが修正されることがあります。

9.3 プロジェクトのプロパティ

プロジェクト ▸ **プロジェクトのプロパティ** のプロジェクトのプロパティウィンドウで、プロジェクト固有のオプションを設定できます。プロジェクト固有のオプションは、上記の **オプション** ダイアログの同じオプションを上書きします。

9.3.1 一般情報プロパティ

 **一般情報** タブの **一般設定** では、以下の設定が可能です：

- プロジェクトファイルの場所を確認する
- プロジェクトのホームフォルダを設定する（ **ブラウザ** パネルの **プロジェクトホーム** アイテムからでも可能です）。パスは、プロジェクトファイルのフォルダに対する相対パス（として入力する）にも、絶対パスにもできます。プロジェクトのホームは、プロジェクトに利用するデータやその他のコンテンツを保存するために使用できます。これは、データセットとプロジェクトファイルが同じ場所には保存されていない場合に便利です。これが入力されていない場合は、 **プロジェクトのホーム** はデフォルトでプロジェクトファイルのフォルダとなります。
- プロジェクトにタイトルを付ける
- 地物が選択されたときに使用する色を選択する
- 背景色を選択する：マップキャンバスに使用する色を選択します
- プロジェクト内でレイヤへのパスを絶対パス（フルパス）として保存するか、プロジェクトファイルの場所に対する相対パスとして保存するかを設定する。レイヤとプロジェクトファイルの両方が移動するもしくは共有する可能性がある場合や、異なるプラットフォーム上のコンピュータからプロジェクトにアクセスする場合には、相対パスの方が良いかもしれません。
- プロジェクトが地図タイルとしてレンダリングされる時のずれを回避するか選択する。このオプションをチェックすると、パフォーマンスの低下につながる可能性があります。
- **属性テーブルの状態を記憶**: プロジェクトでこのオプションにチェックを入れると、開いた任意の属性テーブルはプロジェクト内に保存され、次にそのプロジェクトを開いたときに即座に属性テーブルが復元されます。この機能により、要件に合わせて属性テーブルを特定の設定で構築したプロジェクトを作成し、これらの属性テーブルを再設定するには苦労するような場合には、ワークフローが改善されます。

GIS では面積や距離の計算がしばしば必要になります。しかしながら、これらの値は実際には基礎となる投影設定に関連付けられています。 **計測** フレームではこれらのパラメータを制御できます。以下が選択できます。

- **楕円体** ：距離や面積の計算の基礎となっている楕円体を選択します。これには以下の値があります。
  - **None/Planimetric**：この場合、返される値はデカルト測定値です。このオプションは、新規プロジェクトのデフォルトとして **設定** **オプション** *CRS* **の扱い** メニューから設定することができます
  - **カスタム**：楕円体の長軸と短軸の値を設定する必要があります。
  - または定義済みのリスト（Clarke 1866、 Clarke 1880 IGN、 New International 1967、 WGS 84 など） から既存のものを選択します。
- 長さと周長のための **距離計測の単位** と、 **面積計測の単位** 。これらの設定は、デフォルトでは QGIS オプションで設定した単位が使われますが、現在のプロジェクトの単位で上書きされ、以下の値の単位に使用されます。
  - 属性テーブルのフィールド更新バー
  - フィールド計算機の計算
  - 地物情報表示ツールで表示される長さ、周長および面積値
  - 計測ダイアログに表示されるデフォルトの単位

座標と方位を表示 では、以下の表示に関するカスタマイズができます：

- QGIS のステータスバーの **座標** ボックスや、 地物情報を表示 ツールの結果の **派生した属性** セクションに表示される座標
- マップキャンバスをパン操作した際の方向としてステータスバーに表示される値や、 方位を測る ツールの方位表示

設定可能なパラメータは次の通りです:

- **表示座標の単位** は以下のいずれかです:
  - 地図の単位 は、プロジェクトの CRS に基づきます
  - ジオグラフィック（度）：プロジェクトの CRS が地理的座標系ならばこれに基づき、そうでないならば、これに関連付けられた地理的座標系を使用します。このオプションは例えば地球以外の天体の場合に役立ちます。
  - カスタム投影法単位: 座標表示に任意の CRS を使用できます。
- **座標** *CRS* オプションでは、表示モードに応じて、使用する CRS の確認や定義ができます。
- **座標フォーマット** は 10 進数の角度 、 度、分 、 度、分、秒 として設定でき、さらに以下を表示するかどうかを設定できます：
  - ☐ **方位接頭辞を表示**
  - ☐ **分秒をゼロ埋め表示**
  - ☐ **度をゼロ埋め表示**

- ☐ **ゼロ埋め表示**

- **座標精度** は、小数点以下の桁数を（CRS の種類に応じて）自動とするか、手動で設定できます。
- **座標順** は、座標を CRS の本来の順番で表示する（ デフォルト ）か、 `Easting, Northing` （経度、緯度） または `Northing, Easting` （緯度、経度） の順番に切り替えるかを選べます。
- **方位フォーマット** で利用可能な値は `0` から `180°` （`E/W` 付き） 、 `-180` から `+180°` 、 `0` から `360°` です。 **小数点以下桁数** や、 **ゼロ埋め表示** するかどうかの設定もできます。

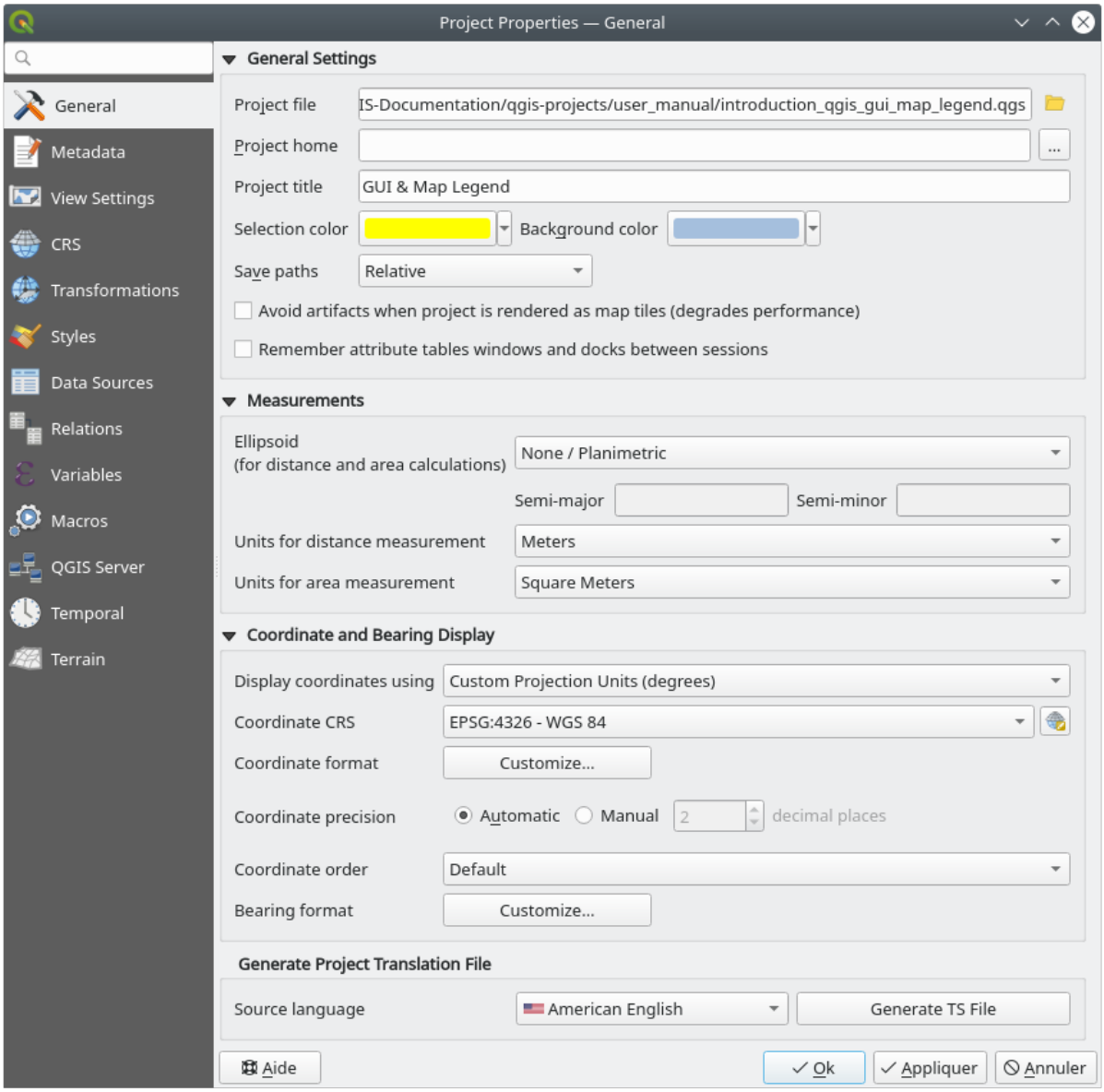


図 9.31: プロジェクトのプロパティダイアログの一般情報タブ

9.3.2 メタデータプロパティ

メタデータ タブでは、制作者、作成日、言語、要約、カテゴリ、キーワード、連絡先の詳細、リンク、履歴など（その他の項目も同様）の詳細なメタデータを定義できます。特定のフィールドが入力されているかどうかを確認する検証機能もありますが、これは強制されません。詳細は、 **ベクタレイヤのメタデータプロパティ** を参照してください。

9.3.3 表示設定

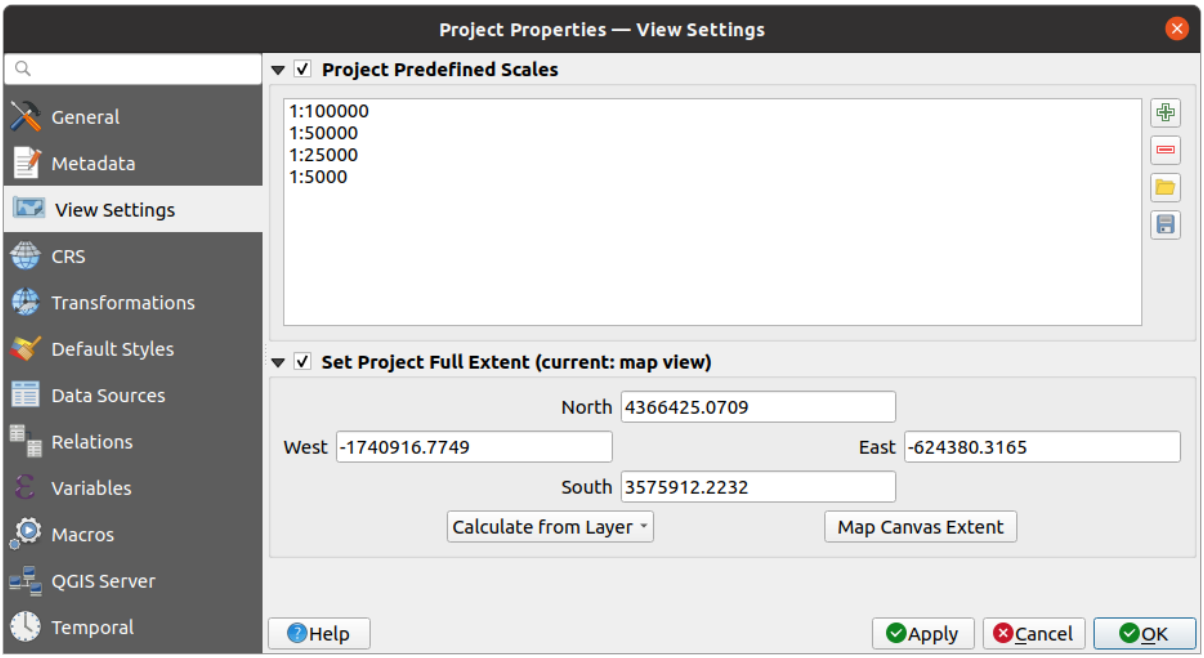


図 9.32: プロジェクトのプロパティダイアログの表示設定タブ

表示設定 タブは、プロジェクトのマップキャンバスをコントロールする手段を提供します。このタブでは、以下の設定ができます：

- **プロジェクト定義済み縮尺** の設定は、ステータスバーの **縮尺** や、レイヤを表示する縮尺の設定、2つ目の 2D マップビューの設定など、縮尺に関連したドロップダウンウィジェットで表示される縮尺のリストです。グローバル設定の **定義済み縮尺** の代わりに使用されます。
- **プロジェクト範囲を設定** ：この範囲は、マップの全域表示でズームする（ ）際に、全レイヤの範囲の代わりに使用されます。この設定は、プロジェクトにウェブ地図のレイヤや国規模・世界規模のレイヤが含まれていても、プロジェクトの実際の対象領域はより狭い地理的範囲である場合に便利です。プロジェクトの全域の座標は、 **範囲セレクタ** ウィジェットを使用して設定できます。

9.3.4 座標参照系（CRS）プロパティ

注釈: QGIS がプロジェクトの投影をどのように扱っているかについての詳細は、 **投影法の利用方法** の専用セクションを読んでください。

座標参照系 タブでは、そのプロジェクトで使用する座標参照系を設定できます。以下の設定があります。

- *CRS* **なし（または未知 / 非地球）**：レイヤの生の座標に基づいてレイヤが描画されます
- 既存の座標参照系で **地理的座標系** 、 **投影された座標系** あるいは **ユーザ定義の座標系** を使用できます。プロジェクトに追加されたレイヤは、重ね合わせるために元の CRS に関係なく、オンザフライでこの CRS に変換されます。

9.3.5 変換プロパティ

変換 タブでは、現在のプロジェクトで適用する測地系変換を設定することで、レイヤの再投影設定を制御することができます。他の設定同様、これは対応するグローバル設定を上書きします。詳細については **測地系変換** を参照してください。

9.3.6 スタイルプロパティ

スタイル タブでは、プロジェクトに固有のシンボルや色を設定でき、異なるマシン間でプロジェクトを安全に共有することができます。

デフォルトのシンボル グループは、既存の `.qml` スタイル定義が無い場合に、新規レイヤがどのように描画されるかを制御します。レイヤのジオメトリタイプに応じて **マーカー** 、 **ライン** 、 **塗りつぶし** に適用される設定や、 **カラーランプ** や **テキストフォーマット** （例えばラベルを有効化した時）のデフォルトを設定できます。これらのアイテムはいずれも、対応するドロップダウンウィジェットの **クリア** エントリからリセットできます。

オプション グループでは、以下の設定ができます：

- デフォルトの **不透明度** を新規レイヤに適用します
- **ランダムな色をシンボルに設定する** は、シンボルの塗りつぶし色を変えることで、すべてのレイヤが同じように描画されるのを避けられます。

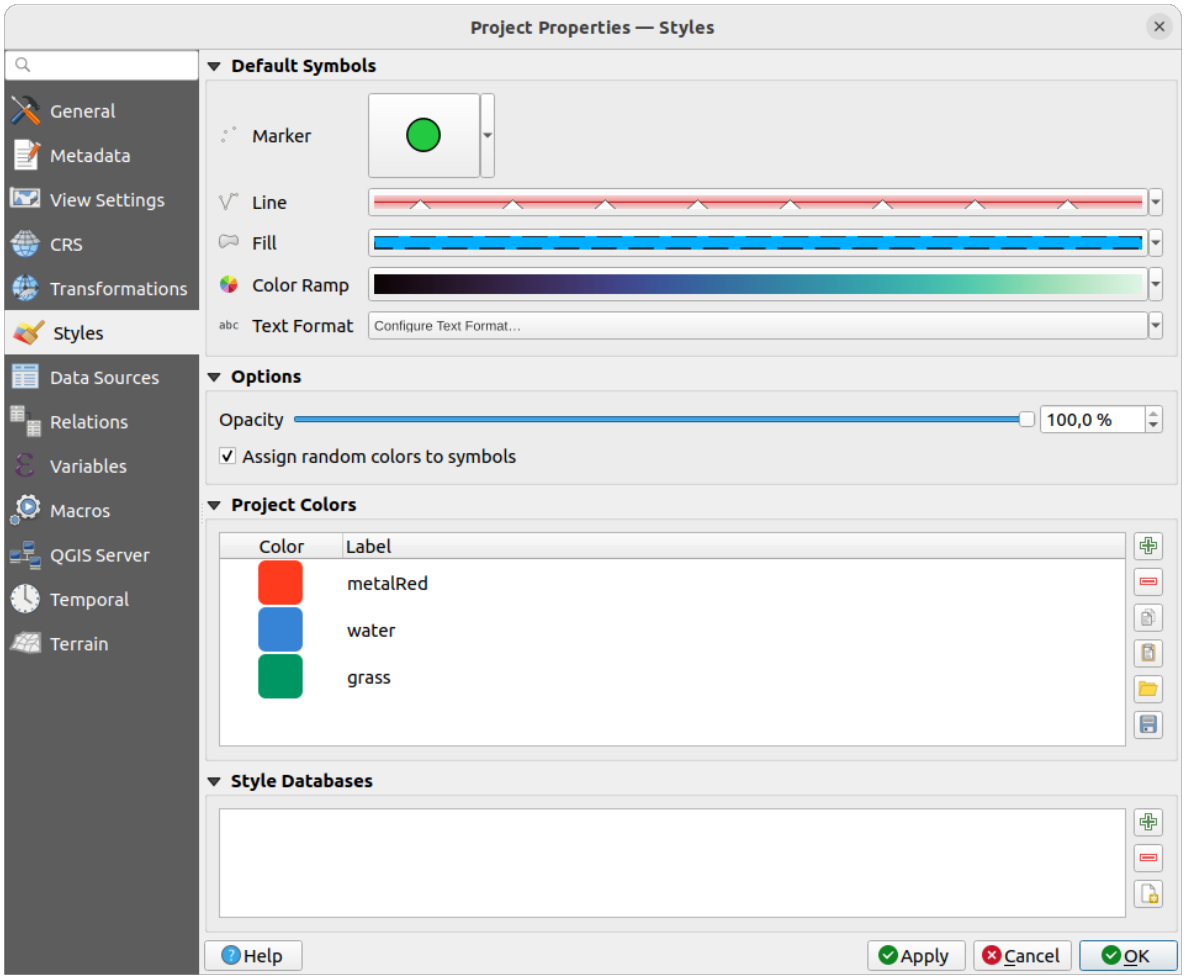


図 9.33: スタイルタブ

また、実行中のプロジェクトの特定の色を定義するセクションもあります。**グローバルの色設定** と同様に、以下の設定ができます。

- **色を追加** あるいは **色を削除**
- **色のコピー** あるいは **色を貼り付け**
- **ファイルから色のインポート** あるいは **色のエクスポート** で色のセットを `.gpl` ファイルからインポート / ファイルへエクスポート

リスト内の色をダブルクリックして **カラーセレクタ** ダイアログで色を調整したり、置き換えたりできます。 **ラベル** 列をダブルクリックして、色の名前を変更することもできます。

これらの色は **プロジェクトの色** として識別され、 **色ウィジェット** の一部としてリストに表示されます。

Tip: プロジェクトの色を使用して、カラーウィジェットを素早く割り当てて更新する

プロジェクトの色はそのラベルを使って参照することができ、それらが使用されているカラーウィジェットはそれらにバインドされます。これによって、多くのプロパティに同じ色を繰り返し設定したり面倒な更新を避けるために、以下のようにすることができます：

1. 色をプロジェクトの色として定義します
2. 設定したい色プロパティの隣にある、 **データによって定義された上書きウィジェット** をクリックします
3. **色** メニューの上にマウスカーソルを置き、プロジェクトの色を選択します。このプロパティには式 `project_color('color_label')` が指定され、カラーウィジェットにはその色が反映されます。
4. ステップ 2 とステップ 3 を必要なだけ繰り返します
5. プロジェクトの色を一度更新するだけで、その色が使われている箇所の「全て」に変更が反映されます。

9.3.7 データソースプロパティ

 **データソース** タブでは、以下の設定ができます：

- **トランザクションモード** は、編集内容がデータプロバイダに送信される方法を定義します。
  - **ローカル編集バッファ**: 編集内容はローカルにバッファされ、レイヤの編集モードを切り替えた時や **編集内容を保存** をクリックした時に送信されます。
  - **自動トランザクショングループ**: これをサポートしているデータソース（Postgres や GeoPackage データベース）では、同一のデータベースに由来する全てのテーブルの編集状態は同期され、サーバーサイドのトランザクションで実行されます。また、編集の変更はローカルでバッファリングするのではなく、直接データベースのトランザクションに送信され、レイヤの編集モードを切り替えた時や **編集内容を保存** をクリックした時に変更がコミットされます。
  - **トランザクショングループのバッファ**: どのデータプロバイダであろうと、編集可能なすべてのレイヤは同期的に切り替えられ、すべての編集はローカルの編集バッファに保存されます。変更の保存は、（プロバイダごとに）すべてのレイヤに対して単一のトランザクションで実行されます。

  このオプションは、プロジェクト内で編集中のレイヤが無い場合にのみ、変更することができることに注意してください。
- **プロバイダ側でデフォルト値を評価する** ：PostgreSQL のテーブルに新しい地物を追加すると、デフォルト値の制約を持つフィールドは、コミット時ではなくフォームを開いた際に値が評価され入力されます。これは、 **地物を追加** フォームのフィールドに `nextval（ 'serial'）` のような式ではなく、期待された値（例えば `25` ）が表示されることを意味します。
- **レイヤの編集状態を記憶**: プロジェクトを保存する時にプロジェクト内で編集中のすべてレイヤは編集中の状態であるとして保存され、プロジェクトを再読み込みした時には、それらのレイヤは即座に編集中の状態になります。
- **レイヤの** *Capabilities* の設定ができます。すなわち、
  - どのレイヤを 情報表示可能 （あるいは不可）とするか、すなわち、 **地物情報表示ツール** に反応するかを設定します。デフォルトでは、レイヤは問合せ可能に設定されています。

- レイヤを 読取専用 として表示するかどうかを設定します。つまり、データプロバイダのケーパビリティに関係なく、ユーザが編集できなくなります。これは弱い保護ですが、ファイルベースのレイヤで作業しているときに、エンドユーザがデータを変更しないようにするための簡単で便利な設定です。
- どのレイヤを 検索可能 とするか、すなわち、 **ロケータウィジェット** を使って問合せ可能かを定義します。デフォルトでは、レイヤは検索可能に設定されています。
- どのレイヤを 必須 とするかを定義します。このリスト内でチェックされたレイヤは、不注意でプロジェクトから削除されないように保護されます。
- どのレイヤを プライベート 、つまりは **レイヤ** パネルでは非表示とするかを定義します。これは、プロジェクトでは必要だが、凡例ツリーや他のレイヤ選択ツールには見せたくない、アクセサリレイヤ（ベースマップ、結合テーブル、値のリレーションのルックアップテーブル、たいていの非空間レイヤなど）のためのものです。表示状態であった場合、そのレイヤはマップキャンバスに表示され、印刷レイアウトの凡例にはレンダリングされます。 **レイヤ** パネル上部のツールバーにある **凡例フィルタ　プライベートレイヤを表示** オプションを使用すると、プライベートレイヤを一時的に表示し、レイヤを操作できるようになります。

レイヤの *Capabilities* テーブルには便利なツールがあります：

- 複数のセルを選択してから **選択切替** を押すと、チェックボックスの状態を変更できます。
- **空間レイヤのみ表示** は、空間レイヤでないものをレイヤのリストから除外します。
- **レイヤのフィルタ**... を使用して、設定したい特定のレイヤを素早く探し出せます。

- **詳細設定** グループでは、 **データソースにメタデータがない場合にプロジェクトを信頼する** かどうかを選択できます。データチェックをスキップすることで、プロジェクトの読み込みが高速化します。これは QGIS Server コンテキストや、巨大なデータベースビューやマテリアライズドビューを持つプロジェクトで便利です。レイヤの範囲は（データソースの代わりに）QGIS プロジェクトファイルから読み込まれ、PostgreSQL プロバイダを使用する場合には、ビューやマテリアライズドビューの主キーの単一性はチェックされません。

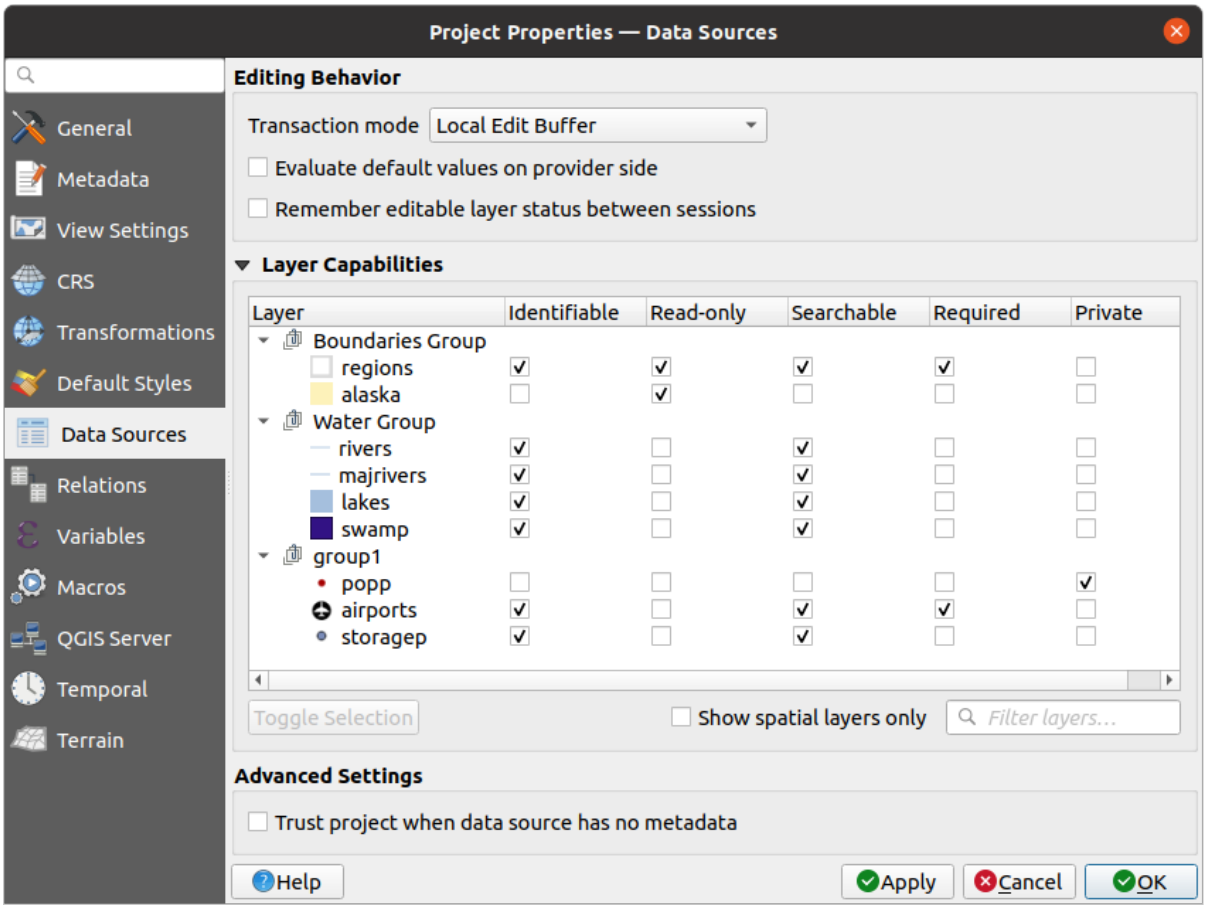


図 9.34: データソースタブ

9.3.8 リレーションプロパティ

リレーション タブは、1:n のリレーションや多態リレーションを定義するために使用されます。リレーションは、プロジェクトのプロパティダイアログで定義されます。あるレイヤにリレーションが存在すると、フォームビューの新しいユーザーインターフェース要素（例えば、地物を特定し、そのフォームを開くとき）に関連するエンティティのリストが表示されます。これは、例えばパイプラインの長さや道路区間の検査履歴を表現するための強力な方法を提供します。1:n リレーションのサポートについての詳細は、*1* **対多または多対多のリレーションの作成** のセクションを参照してください。

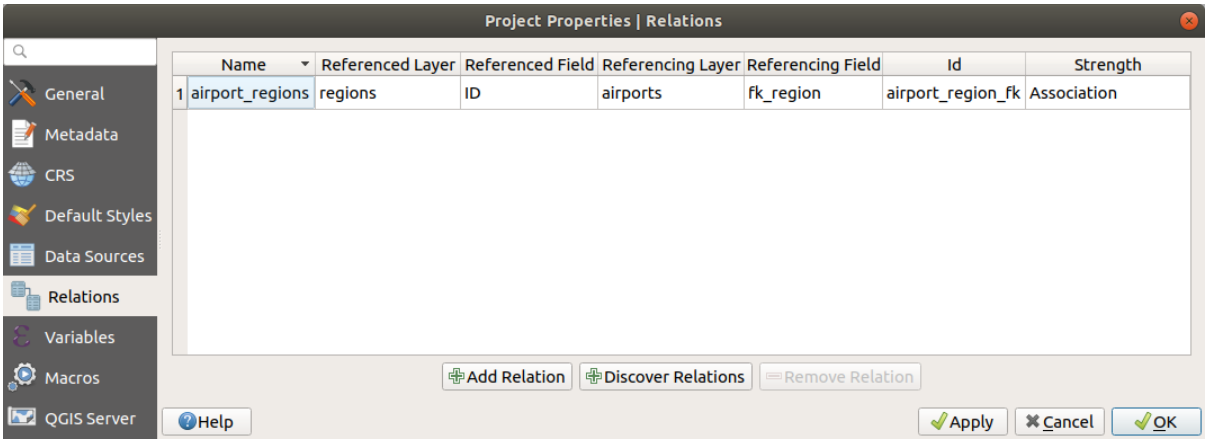


図 9.35: リレーションタブ

9.3.9 変数プロパティ

変数 タブには、プロジェクトレベルで使用可能なすべての変数（すべてのグローバル変数を含む）がリストされています。これに加え、プロジェクトレベルの変数を管理することができます。 ボタンをクリックすると、新しいカスタムプロジェクトレベル変数を追加できます。同様に、リストからカスタムプロジェクトレベルの変数を選択して ボタンをクリックすれば、変数を削除できます。変数の使用法の詳細については、一般ツールの **値を変数に格納する** のセクションを参照してください。

9.3.10 マクロプロパティ

マクロ タブは、プロジェクトの Python マクロの編集のために使います。 現時点では: `openProject()`, `saveProject()` そして `closeProject()` の 3 つのマクロのみ使用可能です。

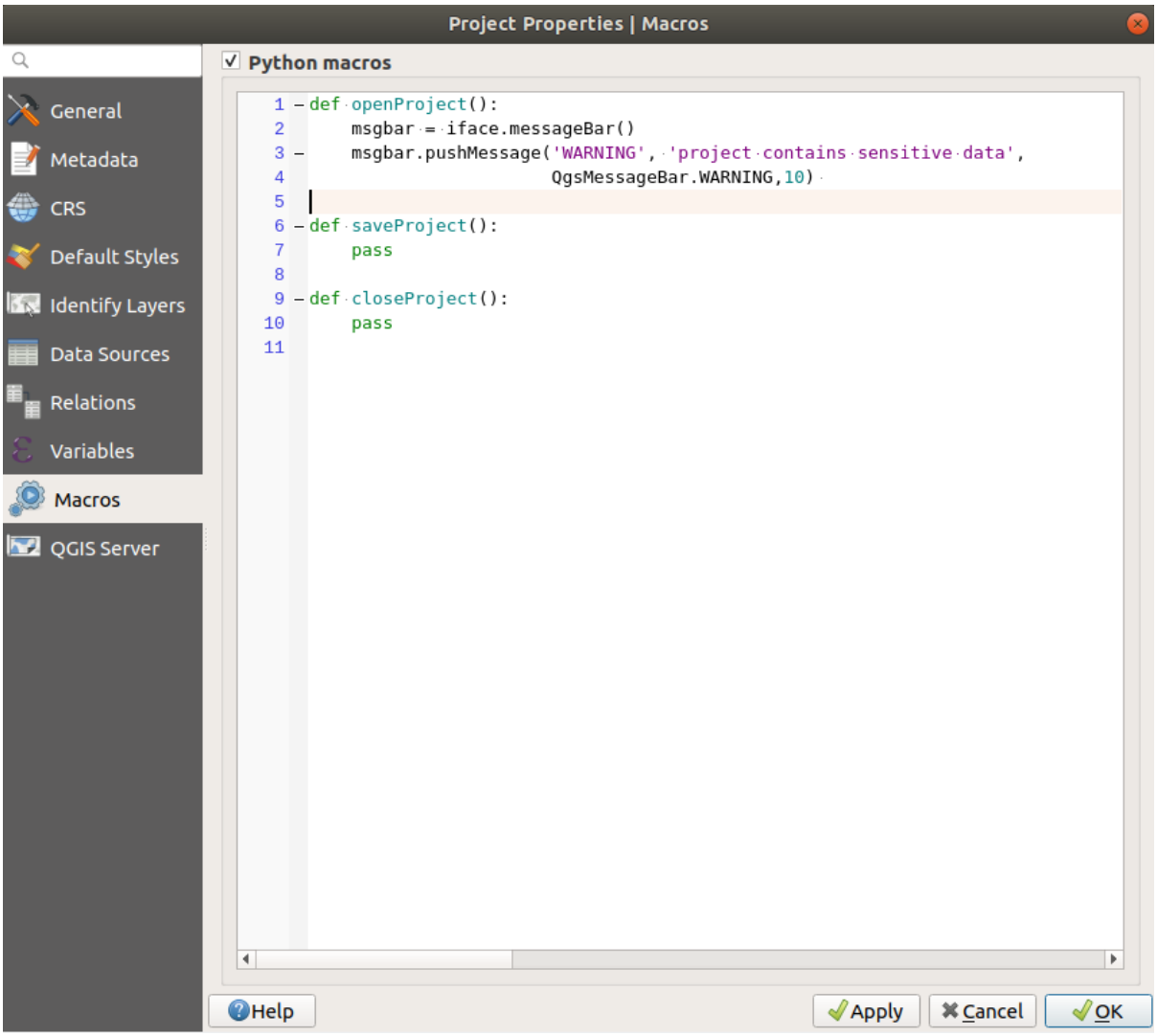


図 9.36: マクロの設定

9.3.11 QGIS サーバー

QGIS **サーバー** タブでは、プロジェクトをオンラインで公開するための設定を行うことができます。ここでは、QGIS サーバーの WMS と WFS のケーパビリティ、範囲、および CRS の制限に関する情報を定義できます。より詳しい情報は Creatingwmsfromproject 以降のセクションを参照してください。

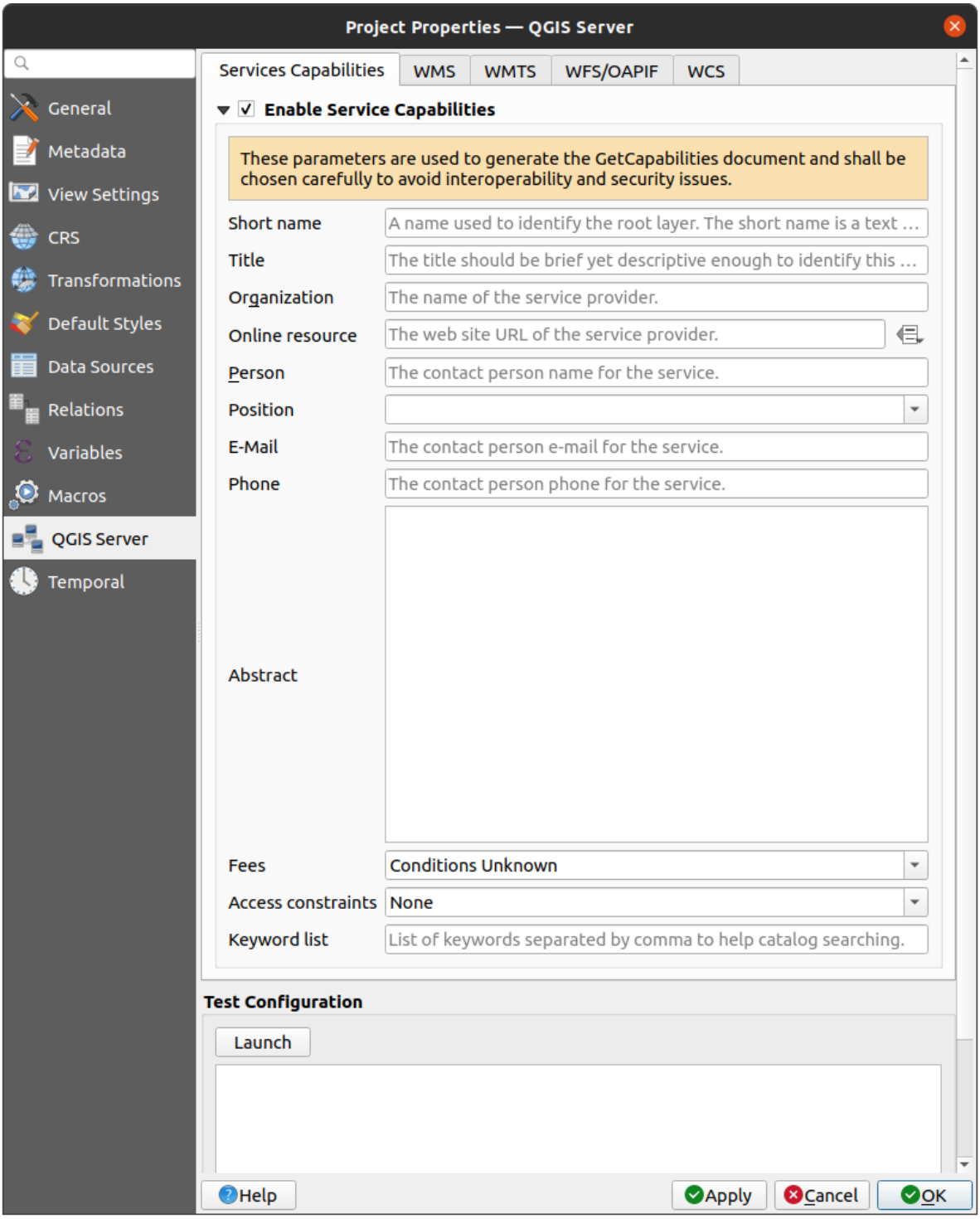


図 9.37: QGIS サーバーの設定

9.3.12 時系列プロパティ

時系列 タブは、プロジェクトの時系列範囲の設定に使用します。時系列範囲は **開始時刻** と **終了時刻** を使用して手入力で設定することも、現在のプロジェクトの時系列レイヤから計算して設定することもできます。プロジェクトの時間範囲は **時系列コントローラパネル** において、マップキャンバスの **時系列ナビゲーション** の管理に使用できます。

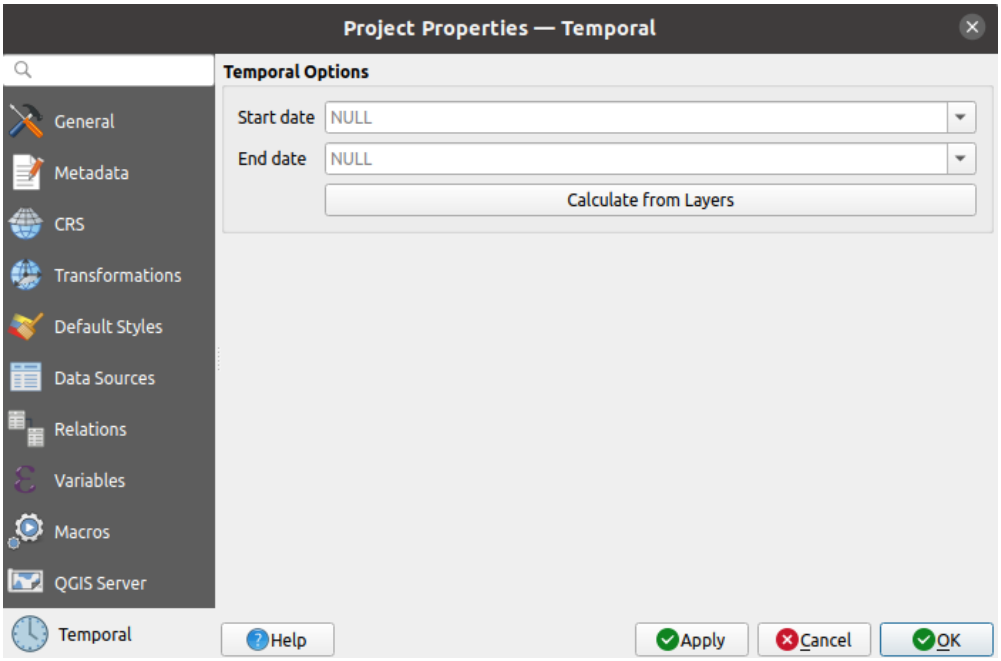


図 9.38: プロジェクトの時系列タブ

9.3.13 地形プロパティ

地形 タブでは、地形や標高のデフォルト設定を行えます。プロジェクトで任意の新規 *3D* **マップ** が作成されると、そのマップはデフォルトでプロジェクト定義の地形設定と同じ設定を使用します。プロジェクトの標高設定も、プロファイルツールで使用されます。

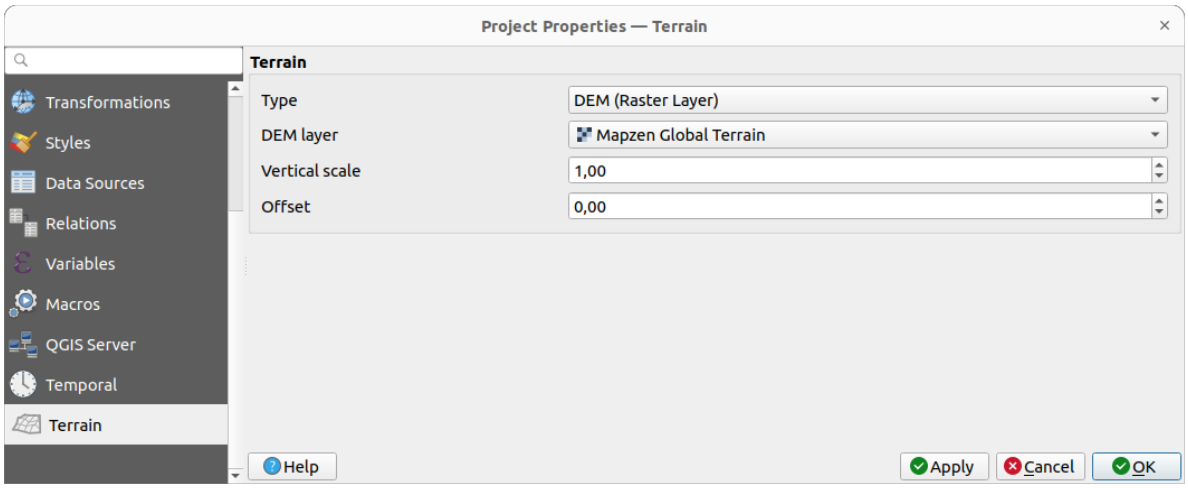


図 9.39: プロジェクトの地形タブ

地形と標高のオプションには、以下のものがあります：

- **平らな地形** では、 **地形の高さ** の設定
- *DEM* **ラスタ** では、 *DEM* **レイヤ** とバンド値に適用する **鉛直スケール** の係数、鉛直方向の **オフセット** の定義
- **メッシュ** では、 **メッシュレイヤ** と頂点の Z 値に適用する **鉛直スケール** の係数、鉛直方向の **オフセット** の定義

これらの設定は、3D マップの **設定ダイアログ** から上書きできます。

9.4 インタフェースのカスタマイズ

インタフェースのカスタマイズ ダイアログでは、QGIS のユーザーインタフェースのほぼすべての要素を有効（無効）にできます。これは、必要なアイコン、メニュー、またはパネルのみが含まれた「軽量な」バージョンの QGIS をエンドユーザーに提供したい場合に非常に便利です。

注釈: 変更が適用される前に、QGIS を再起動する必要があります。

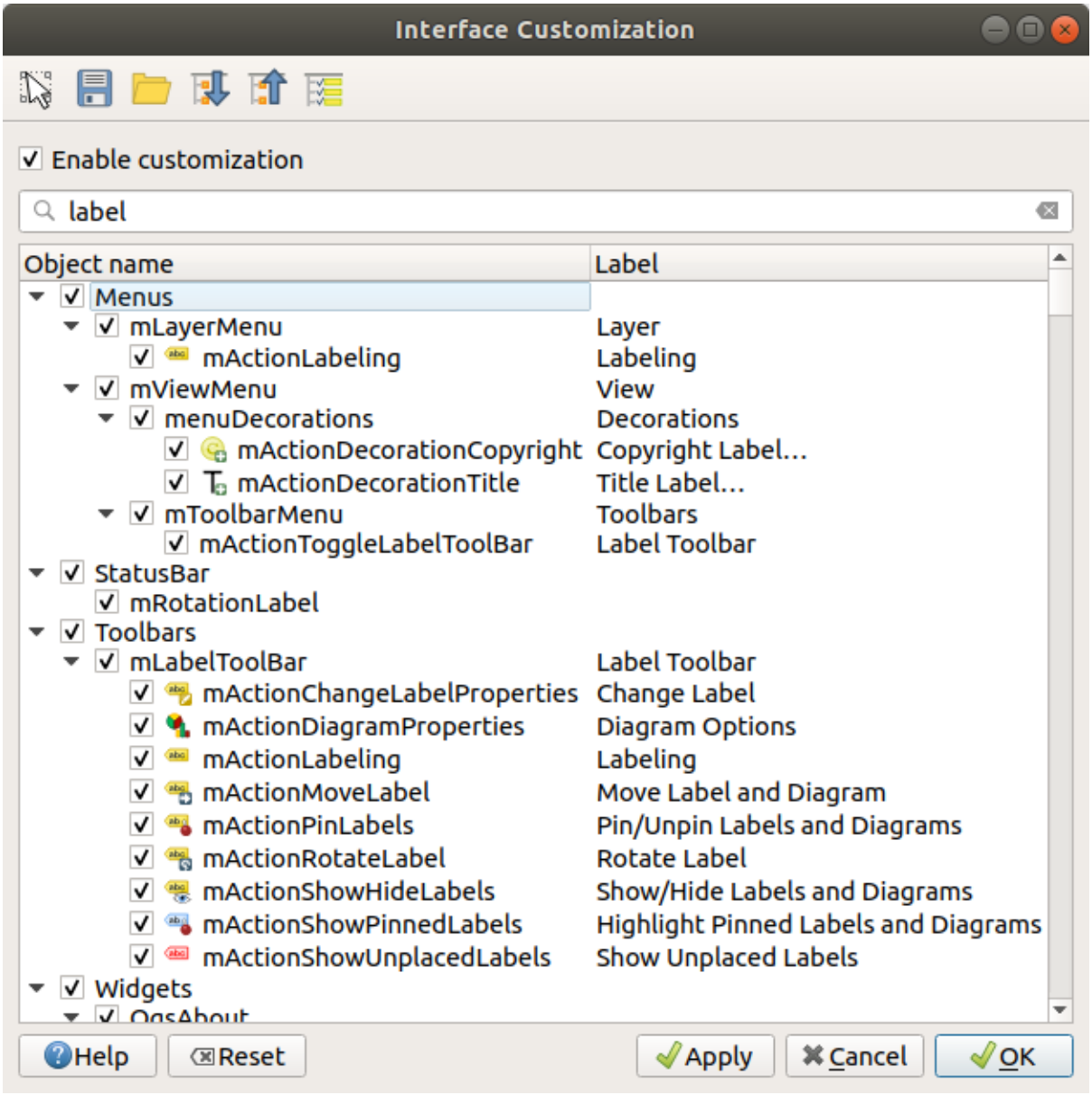


図 9.40: カスタマイズダイアログ

カスタマイズを有効にする チェックボックスにチェックを入れることは、QGIS のカスタマイズへの第一歩です。これによりツールバーとウィジェットパネルが有効になり、いくつかの GUI アイテムのチェックを外して無効にすることができます。

設定可能な項目には以下のものがあります。

- **メニューバー** にある **メニュー** やそのサブメニュー
- **パネル** の全て（ **パネルとツールバー** 参照）
- **ステータスバー** で説明されている **ステータスバー** やその項目
- **ツールバー** : バー全体やアイコン
- ラベル、ボタン、コンボボックス等の、QGIS のダイアログにある **ウィジェット**

を使用すると、QGIS インタフェース内で非表示にしたい項目をクリックしたときに自動的にカスタマイズダイアログで対応する項目のチェックを外します。また、 **検索** ボックスを使用して、項目の名前やラベルから項目を検索することもできます。

設定を行ったら、 **適用** または *OK* をクリックして変更内容を確認します。この設定は、次回の起動時に QGIS がデフォルトで使用するものになります。

変更は 

ボタン使用して `.ini` ファイルにも保存できます 。これは、複数のユーザー間で共通の QGIS インターフェイスを共有するための便利な方法です。 `.ini` ファイルをインポートするには、共有先のコンピュータで 

をクリックするだけです。また同様に、 **コマンドラインツール** を実行してさまざまなユースケースに応じた各種設定を保存できます。

Tip: 定義済みの QGIS を簡単に復元する

QGIS の GUI の初期構成は、以下のいずれかの方法で復元できます。

- カスタマイズダイアログで  **カスタマイズを有効にする** オプションのチェックを外すか、 

ボタンをクリックする
- **設定** **オプション** メニューの **システム** タブ内、 **設定** フレームにある **リセット** ボタンを押す
- コマンドプロンプトで次のコマンドラインを使用して QGIS を起動する：`qgis --nocustomization`
- **設定** **オプション** メニューの **詳細設定** タブ（ **警告** 参照）内で、 *UI* *Customization* *Enabled* 変数の値を `false` に設定する

ほとんどの場合、変更を適用するためには QGIS を再起動する必要があります。

9.5 キーボードショートカット

QGIS には多数の機能のためのデフォルトのキーボードショートカットが用意されています。これらは **メニューバー** のセクションにあります。さらに、 **設定**  **キーボードショートカット**... のメニューオプションを使用すると、デフォルトのキーボードショートカットを変更したり、QGIS の機能に新しいショートカットを追加したりすることができます。

図 9.41: キーボードショートカットの定義オプション

設定するのは非常に簡単です。ダイアログの上部にある検索ボックスを使用して特定のアクションを見つけ、リストからそれを選択して以下のボタンをクリックします：

- **変更** ボタンを押し、新しいショートカットとして割り当てる新しい組み合わせを押す
- **設定解除** ボタンを押し、割り当てられているショートカットを解除する
- **デフォルトに設定** ボタンを押し、ショートカットをもとのデフォルト値に戻す

カスタマイズしたいツールについて、上記の手順に従ってショートカットを設定してください。設定が完了したら、ダイアログを **閉じる** と変更が適用されます。また、変更をユーザーショートカットのみ、または全ショートカットを `.XML` ファイルとして **保存** したり、全てのショートカットを `.PDF` ファイルとして保存し、それらを他の QGIS インストール環境に **読み込み** させることもできます。

9.6 高度な設定で QGIS を実行する

9.6.1 コマンドラインと環境変数

QGIS **の起動** は OS 上の他のアプリケーションと同様に実行されることを見てきました。 QGIS では、より高度な使用例のためにコマンドラインオプションを提供しています（場合によっては、コマンドラインオプションの代わりに環境変数を使用できます）。オプションのリストを取得するには、コマンドラインで `qgis --help` と入力します：

```
QGIS is a user friendly Open Source Geographic Information System.
Usage: /usr/bin/qgis.bin [OPTION] [FILE]
  OPTION:
        [-v, --version]     display version information and exit
        [-s, --snapshot filename]   emit snapshot of loaded datasets to given file
        [-w, --width width] width of snapshot to emit
        [-h, --height height]       height of snapshot to emit
        [-l, --lang language]       use language for interface text (changes existing␣
↪override)
        [-p, --project projectfile] load the given QGIS project
        [-e, --extent xmin,ymin,xmax,ymax]  set initial map extent
        [-n, --nologo]      hide splash screen
        [-V, --noversioncheck]      don't check for new version of QGIS at startup
        [-P, --noplugins]   don't restore plugins on startup
        [-B, --skipbadlayers]     don't prompt for missing layers
        [-C, --nocustomization]     don't apply GUI customization
        [-z, --customizationfile path]      use the given ini file as GUI␣
↪customization
        [-g, --globalsettingsfile path]     use the given ini file as Global Settings␣
↪(defaults)
        [-a, --authdbdirectory path] use the given directory for authentication␣
↪database
        [-f, --code path]   run the given python file on load
        [-d, --defaultui]   start by resetting user ui settings to default
        [--hide-browser]        hide the browser widget
        [--dxf-export filename.dxf]     emit dxf output of loaded datasets to given␣
↪file
        [--dxf-extent xmin,ymin,xmax,ymax]      set extent to export to dxf
        [--dxf-symbology-mode none|symbollayer|feature] symbology mode for dxf output
        [--dxf-scale-denom scale]       scale for dxf output
        [--dxf-encoding encoding]       encoding to use for dxf output
        [--dxf-map-theme maptheme]      map theme to use for dxf output
        [--take-screenshots output_path]        take screen shots for the user␣
↪documentation
        [--screenshots-categories categories]   specify the categories of screenshot␣
↪to be used (see QgsAppScreenShots::Categories).
        [--profile name]        load a named profile from the user's profiles folder.
```

(次のページに続く)

(前のページからの続き)

```
       [-S, --profiles-path path]  path to store user profile folders. Will create␣
↪profiles inside a {path}\profiles folder
       [--version-migration]   force the settings migration from older version if␣
↪found
       [--openclprogramfolder]         path to the folder containing the sources for␣
↪OpenCL programs.
       [--help]                this text
       [--]            treat all following arguments as FILEs

 FILE:
   Files specified on the command line can include rasters,
   vectors, and QGIS project files (.qgs and .qgz):
    1. Rasters - supported formats include GeoTiff, DEM
       and others supported by GDAL
    2. Vectors - supported formats include ESRI Shapefiles
       and others supported by OGR and PostgreSQL layers using
       the PostGIS extension
```

Tip: コマンドライン引数を利用する例

コマンドラインで 1 つまたは複数のデータファイルを指定して QGIS を起動できます。たとえば、qgis_sample_data ディレクトリにいると仮定すると、次のコマンドを使用して、起動時にベクタレイヤとラスタファイルをロードするように設定された QGIS を起動できます： `qgis ./raster/landcover.img ./gml/lakes.gml`

--version

このオプションは QGIS のバージョン情報を返します。

--snapshot

このオプションを使うと PNG 形式で現在のビューのスナップショットを作れます。これは、多数のプロジェクトがあり、データからスナップショットを作成したい場合、または更新されたデータで同じプロジェクトのスナップショットを作成したい場合に便利です。

このオプションを使うと 800x600 ピクセルの PNG ファイルが作成されます。 `--width` と `--height` を引数に加えることでサイズの調整ができます。 `--snapshot` の後にファイル名を指定できます。例えば:

```
qgis --snapshot my_image.png --width 1000 --height 600 --project my_project.qgs
```

--width

このオプションは、出力されるスナップショットの幅を返します（ --snapshot と共に使用されます)。

--height

このオプションは、出力されるスナップショットの高さを返します（ --snapshot と共に使用されます)。

--lang

ロケールに基づいて、QGIS は正しい言語対応を選択します。言語を変更したいならば言語コードを指定できます。例えば、 qgis --lang it であればイタリア語対応で QGIS を起動します。

--project

既存のプロジェクトファイルから QGIS を起動することも可能です。プロジェクト名の後に --project コマンドラインオプションを追加するだけで、指定されたファイル内のすべてのレイヤが読み込まれた状態で QGIS が開きます。

--extent

ある地図の領域を指定して QGIS を起動する場合はこのオプションを使います。この場合、下記のようにカンマで区切られた書式の領域指定で領域を包含する長方形を指定する必要があります。

```
--extent xmin,ymin,xmax,ymax
```

このオプションは -project オプションと組み合わせて、特定のプロジェクトを目的の範囲で開くようにすれば、より意味のあるものになるでしょう。

--nologo

このオプションは、QGIS の起動時のスプラッシュスクリーンを非表示にします。

--noversioncheck

起動時の QGIS の新しいバージョンの検索をスキップします。

`--noplugins`

起動時にプラグインのトラブルがある場合、起動時にプラグインのロードを無効にできます。プラグインは、後からプラグインマネージャで有効にすることができます。

`--nocustomization`

このオプションを使用すると、既存の *GUI* **カスタマイズ** は起動時に適用されません。これは、非表示のボタン、メニュー項目、ツールバーなどが QGIS の起動時に表示されることを意味します。これは永続的な変更ではありません。このオプションなしで QGIS を起動すると、カスタマイズが再度適用されます。

このオプションは、カスタマイズによって削除されたツールへのアクセスを一時的に許可する場合に役立ちます。

`--skipbadlayers`

このオプションを使用すると、スタートアップ時に QGIS が **利用できないレイヤを処理** ダイアログを表示させないようにできます。プロジェクトファイルが読み込まれると、見つからないレイヤは利用できないレイヤとして保持されます。このトピックの詳細については、 **壊れたファイルパスの取り扱い** を参照してください。

`--customizationfile`

このオプションを使用すると、起動時に使用される UI カスタマイズファイルを定義できます。

`--globalsettingsfile`

これと同等な環境変数は、 `QGIS_GLOBAL_SETTINGS_FILE` です。

このオプションを使用して、デフォルト設定とも呼ばれるグローバル設定ファイル（ `.ini` ）のパスを指定できます。指定されたファイルの設定は、元のインラインのデフォルト設定を置き換えますが、ユーザープロファイルの設定はそれらを上書きして設定されます。

QGIS はデフォルトのグローバル設定ファイルを以下の順番で検索し、最初に見つかったものを使用します：

- コマンドラインパラメータで指定されたパス
- 環境変数で定義されたパス
- アプリケーションのデータ（AppDataLocation）フォルダ。ここには永続的なアプリケーションデータを格納できます。これはユーザーまたはシステム管理者が管理し、インストーラによる変更は行われず、コマンドラインパラメータや設定環境変数を渡すなどの追加設定を必要としません。このフォルダは OS によって異なります：
  - `$HOME/.local/share/QGIS/QGIS3/`
  - `C:\Users\<username>\%AppData%\Roaming\QGIS\QGIS3\`

  - X $HOME/Library/Application Support/QGIS/QGIS3/

- インストールディレクトリ。すなわち、 your_QGIS_package_path/resources/qgis_global_settings.ini

現在、設定を書き込むファイルを指定する方法はありません。したがって、元の設定ファイルのコピーを作成し、名前を変更し、適応させることができます。

qgis_global_setting.ini ファイルパスをネットワーク共有フォルダに設定すると、システム管理者は 1 つのファイルを編集するだけで、複数のマシンのグローバル設定とデフォルトを変更できます。

--authdbdirectory

このオプションは --globalsettingsfile に似ていますが、認証データベースが保存およびロードされるディレクトリへのパスを定義します。

--code

このオプションを使用すると、指定した python ファイルを QGIS を開始した直後に実行します。

例えば、以下の内容の load_alaska.py という名前の python ファイルがあるとします。

```
from qgis.utils import iface
raster_file = "/home/gisadmin/Documents/qgis_sample_data/raster/landcover.img"
layer_name = "Alaska"
iface.addRasterLayer(raster_file, layer_name)
```

ファイル load_alaska.py が配置されているディレクトリにいると仮定して、次のコマンドを使用することで、QGIS を起動し、ラスタファイル landcover.img をロードし、レイヤに 'Alaska' という名前を付けることができます。

```
qgis --code load_alaska.py
```

--defaultui

ロード時に、ユーザーインタフェース（UI）を既定の設定に **永続的にリセット** します。このオプションは、パネルとツールバーの表示、位置、サイズを復元します。再度変更しない限り、次のセッションではデフォルトの UI 設定が使用されます。

このオプションは *GUI* **のカスタマイズ** には何の影響もないことに注意してください。GUI のカスタマイズによって隠されたアイテム（例えばステータスバー）は、 --defaultui オプションを使ったとしても隠されたままです。--nocustomization オプションについても参照してください。

`--hide-browser`

ロード時に、ユーザーインタフェースから **ブラウザ** パネルを非表示にします。パネルを有効にするには、ツールバーのスペースを右クリックするか、 **表示** → **パネル** （ Linux KDE では **設定** → **パネル** ）を使用します。

ブラウザパネルが再度有効にならない限り、次のセッションでもパネルは隠されたままになります。

`--dxf-*`

これらのオプションは、DXF ファイルに QGIS プロジェクトをエクスポートするために使用できます。いくつかのオプションが用意されています：

- *--dxf-export* : レイヤを出力する DXF ファイル名;
- *--dxf-extent* : 最終的な DXF ファイルの範囲;
- *--dxf-symbology-mode* ：ここでは、none（シンボルなし）、symbollayer（シンボルレイヤシンボル）、feature（地物シンボル）の値を使用できます。
- *--dxf-scale-denom*: シンボルの縮尺分母;
- *--dxf-encoding*: ファイルのエンコーディング
- *--dxf-map-theme*: レイヤツリー設定から **地図テーマ** を選択

`--take-screenshots`

ユーザドキュメントのスクリーンショットを撮ります。 `--screenshots-categories` とともに使用して、ドキュメントスクリーンショットのどのカテゴリ/セクションを作成するかをフィルターできます（ QgsAppScreenShots::Categories 参照）。

`--profile`

ユーザのプロファイルフォルダから特定のプロファイルを使用して QGIS を読み込みます。変更しない限り、選択したプロファイルは次の QGIS セッションで使用されます。

`--profiles-path`

このオプションを使用すると、プロファイル（ユーザ設定）をロードして保存するパスを選択できます。`{path}\profiles` フォルダ内にプロファイルを作成します。このフォルダには、設定、インストールされたプラグイン、プロセシングモデルとスクリプトなどが含まれます。

このオプションを使用すると、たとえば、すべてのプラグインと設定をフラッシュドライブで持ち運んだり、ファイル共有サービスを使用して異なるコンピュータ間で設定を共有したりできます。

これと同等な環境変数は、`QGIS_CUSTOM_CONFIG_PATH` です。

`--version-migration`

古いバージョンの設定（**例えば**、 QGIS 2.18 の `.qgis2` フォルダ）が見つかったときに、このオプションはデフォルトの QGIS プロファイルに設定をインポートします。

`--openclprogramfolder`

このオプションを使用すると、OpenCL プログラムの代替パスを指定できます。これは、既存のプログラムを置き換えることなく、プログラムの新しいバージョンをテストする開発者にとって便利です。

同等の環境変数は `QGIS_OPENCL_PROGRAM_FOLDER` です。

9.6.2 組織内での QGIS の導入

カスタム構成ファイルを使用して組織内に QGIS をデプロイする必要がある場合、最初に `your_QGIS_package_path/resources/qgis_global_settings.ini` にあるデフォルト設定ファイルの内容をコピーして貼り付ける必要があります。このファイルには、 `[]` で始まるブロックで識別されるデフォルトのセクションがすでに含まれています。これらのデフォルト値はそのままにしておき、ファイルの下部に独自のセクションを追加することをお勧めします。ファイル内でセクションが重複している場合、QGIS は上から下に向かって検索して見つかった最後のセクションを使用します。

QGIS のバージョンチェックを無効にするには `allowVersionCheck=false` と変更します。

新規インストール後に移行ウィンドウを表示したくない場合は、次のセクションが必要です。

```
[migration]
fileVersion=2
settings=true
```

グローバルスコープのカスタム変数を追加したい場合は、以下が必要です。

```
[variables]
organisation="Your organization"
```

`INI` ファイルの設定による可能性について理解するため、QGIS デスクトップで設定を行った後、プロファイルにある `INI` ファイルをテキストエディタを使って検索することをお勧めします。 WMS/WMTS、PostGIS 接続、プロキシ設定、地図の tips など、数多くの設定を `INI` ファイルを使用して行うことができます。

最後に、カスタマイズしたファイルのパスを環境変数 `QGIS_GLOBAL_SETTINGS_FILE` に設定する必要があります。

さらに、Python マクロ、カラーパレット、レイアウトテンプレート、プロジェクトテンプレートなどのファイルを QGIS システムディレクトリまたは QGIS ユーザープロファイルのいずれかに配置することもできます。

- レイアウトテンプレートは `composer_templates` ディレクトリ内に配置する必要があります。
- プロジェクトテンプレートは `project_templates` ディレクトリ内に配置する必要があります。

- カスタム Python マクロは python ディレクトリ内に配置する必要があります。

第10章　投影法の利用方法

座標参照系（CRS）は、数値座標を地球の表面上の位置に関連付ける方法です。 QGIS では、それぞれに異なる使用例や長所・短所がある約 7,000 の標準 CRS に対応しています！ QGIS プロジェクトやデータに対して適切な参照系を選択するのは複雑な作業になることがありますが、幸いなことに QGIS ではこの選択の手引きとなり、さまざまな CRS をできるだけ透過的かつ正確に使用できるようにしています。

10.1 投影法サポートの概要

QGIS では約 7,000 の既知の CRS をサポートしています。これらの標準 CRS は、欧州石油探査グループ（European Petroleum Search Group：EPSG）およびフランス国土地理院（Institut Geographique National de France：IGNF）によって定義されたものに基づいており、基礎となる「Proj」投影ライブラリを通じて QGIS で利用できます。通常、これらの標準的な投影法は、組織：コードの組み合わせを使用して識別されます。ここで、組織は「EPSG」や「IGNF」などの組織名であり、コードは特定の CRS に関連付けられた一意の番号です。たとえば、一般的な WGS 84 緯度/経度 CRS は識別子「EPSG：4326」で知られており、Web マッピング標準 CRS は「EPSG：3857」です。

ユーザが作成したカスタム CRS は、ユーザ CRS データベースに保存されます。カスタム座標参照システムの管理については **カスタム座標参照系** のセクションを参照してください。

10.2 レイヤの座標参照系

データを特定の対象 CRS に正しく投影するためには、データに座標参照系に関する情報が含まれているか、レイヤに正しい CRS を手動で割り当てる必要があります。PostGIS レイヤの場合、QGIS はその PostGIS レイヤが作成されたときに指定された空間参照識別子を使用します。GDAL でサポートされているデータについては、QGIS は CRS を指定するための認識された手段の存在に依存します。例えば、シェープファイル形式の場合、これはレイヤの CRS の ESRI Well-Known Text (WKT) 表現を含むファイルです。この投影ファイルは `.shp` ファイルと同じベース名で、 `.prj` という拡張子を持ちます。例えば、 `alaska.shp` は、`alaska.prj` という名前の対応する投影ファイルを持つことになります。

QGIS にレイヤが読み込まれるたびに、QGIS はそのレイヤの正しい CRS を自動的に判断しようと試みます。ただし、CRS 情報を持たずにレイヤが読み込まれた場合など、 場合によっては自動判別が不可能な場合もあります。レイヤの正しい CRS を自動的に判断できない場合の QGIS の動作を以下の手順で設定することができます:

1. **設定**　🔧**オプション...**　**座標参照系（*CRS*）** を開く

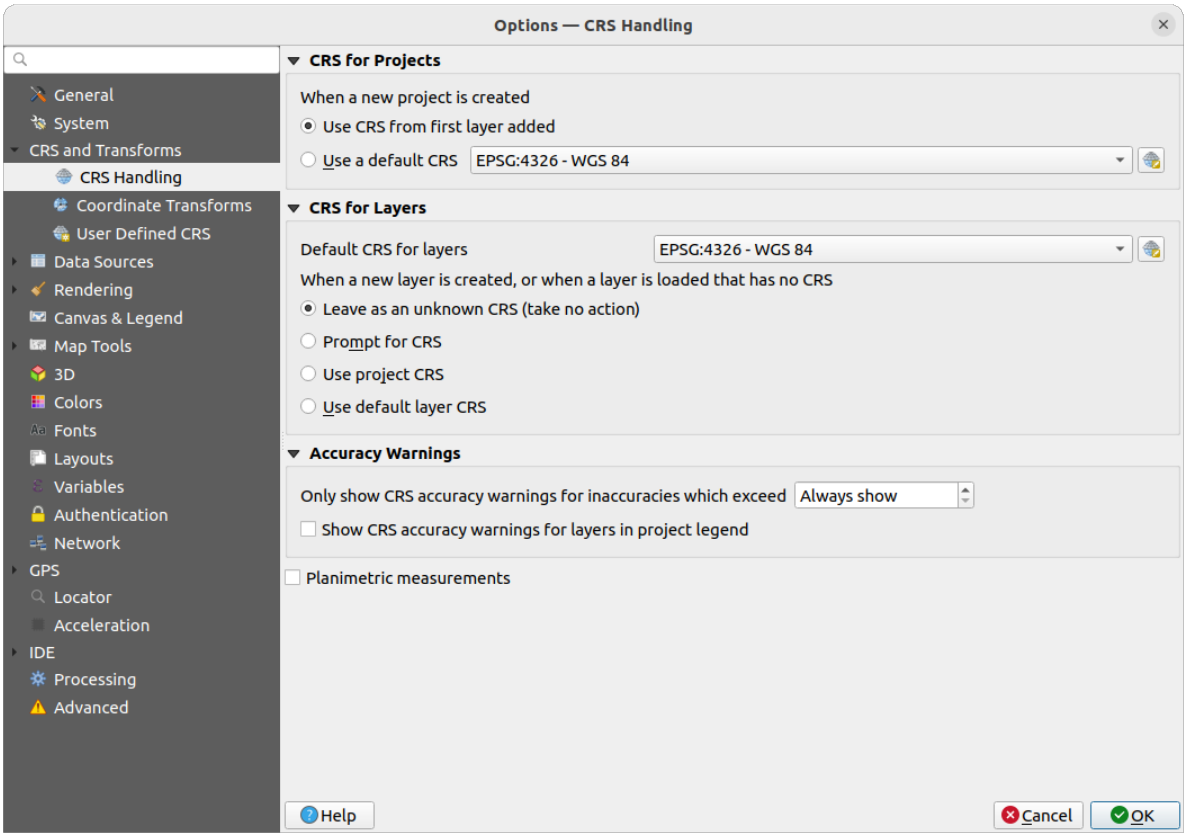


図 10.1: QGIS オプションダイアログの座標参照系（CRS）タブ

2. **レイヤの** *CRS* グループで、**新規レイヤが作成されるか、あるいは** *CRS* **のないレイヤが読み込まれた場合**... のアクションとして、以下のうちどれかを設定します。

   - **未知の** *CRS* **のまま**: CRS を持たないレイヤがロードされた時に CRS を選択するプロンプトを表示せず、CRS の選択は後回しにされます。一度にたくさんのレイヤを読み込むときに便利です。CRS の選択が後回しになっているレイヤは、**レイヤ** パネル内でレイヤの横にある アイコンで識別可能です。これらのレイヤは座標参照されず、レイヤの座標は純粋な数値の非地球投影の値として扱われます。これはつまり **プロジェクトの** *CRS* **が「投影なし」に設定された場合** と同じ動作です。
   - *CRS* **ダイアログを表示**: CRS を手動で選択するよう、ダイアログが表示されます。正しい選択をすることが非常に重要です。なぜなら、間違った CRS を選択をすると、レイヤが地表の間違った位置に置かれてしまいます！付随するメタデータにレイヤの正しい CRS が記述されていることもありますが、場合によっては、使用すべき正しい CRS を決定するためにデータの元の作成者に連絡する必要があるでしょう。
   - **プロジェクト** *CRS* **を使う**
   - **デフォルト** *CRS* **を使う** フレーム上部にある、**レイヤのデフォルト** *CRS* コンボボックスで設定されたものを使います。

Tip: CRS のない、あるいは CRS を間違った複数のレイヤに対して、同じ CRS を一度の操作で設定するには、

1. **レイヤ** パネルで複数のレイヤを選択します
2. `Ctrl+Shift+C` を押します。あるいは選択したレイヤのうち一つを右クリックするか、**レイヤ　レイヤ** *CRS* **の設定** メニューをクリックします。
3. 使用する正しい CRS を探し、選択します
4. *OK* ボタンを押します。レイヤのプロパティダイアログの **ソース** タブで、CRS が正しく設定されていることを確認できます。

この設定で CRS を変更しても、基礎となっているデータソースは全く変更されないことに留意してください。QGIS がレイヤの生座標を現在の QGIS プロジェクト内においてどのように解釈するかを変更するだけです。

10.3 プロジェクトの座標参照系

QGIS の各プロジェクトも、プロジェクトに関連づけられた座標参照系を持っています。プロジェクトの CRS は、データがその基礎となる生の座標から QGIS のマップキャンバス内にレンダリングされる地図平面に投影される方法を決定します。

QGIS はラスタデータとベクタデータの両方に対して「オンザフライ」の CRS 変換をサポートしています。これは、プロジェクト内の特定のマップレイヤの本来の CRS に関係なく、レイヤの CRS を常にプロジェクト用に定義された共通の CRS に自動的に変換します。舞台裏では、QGIS はプロジェクト内に含まれるすべてのレイヤを透過的にプロジェクトの CRS に再投影します。これによって、全てのレイヤがお互いに対して正しい位置にレンダリングされるのです！

QGIS プロジェクトに適切な CRS を選択することが大切です。不適切な CRS を選択すると、地図が歪んで表示されたり、現実世界の地物の相対的な大きさや位置が正しく反映されない可能性があります。通常、狭い地理的領域での作業では、特定の国や行政区域内で使用される標準的な CRS が多数存在します。図化する地域にどの CRS が適当か、あるいは標準的な選択であるかを調査し、QGIS プロジェクトがこれらの標準に従っていることを確認することが重要です。

デフォルトでは、QGIS はグローバルなデフォルトの投影法を使用して新しいプロジェクトを開始します。このデフォルトの CRS は `EPSG:4326`（「WGS 84」としても知られています）であり、グローバルな緯度/経度ベースの参照系です。このデフォルト CRS は **設定**　🔧 **オプション**...（ 図 10.1 を参照）の **座標参照系（***CRS***）** タブで、**新しいプロジェクトが作られた時の** *CRS* の設定で変更することができます。 プロジェクトの CRS を新規プロジェクトに読み込まれた最初のレイヤの CRS に合わせて自動的に設定するオプションや、新規作成されたすべてのプロジェクトで使用するデフォルトの CRS を選択するオプションがあります。この選択はその後の QGIS セッションで使用するために保存されます。

プロジェクトの CRS は **プロジェクト　プロパティ**... ダイアログの **座標参照系** タブからも設定できます。また、QGIS のステータスバーの右下にも表示されます。

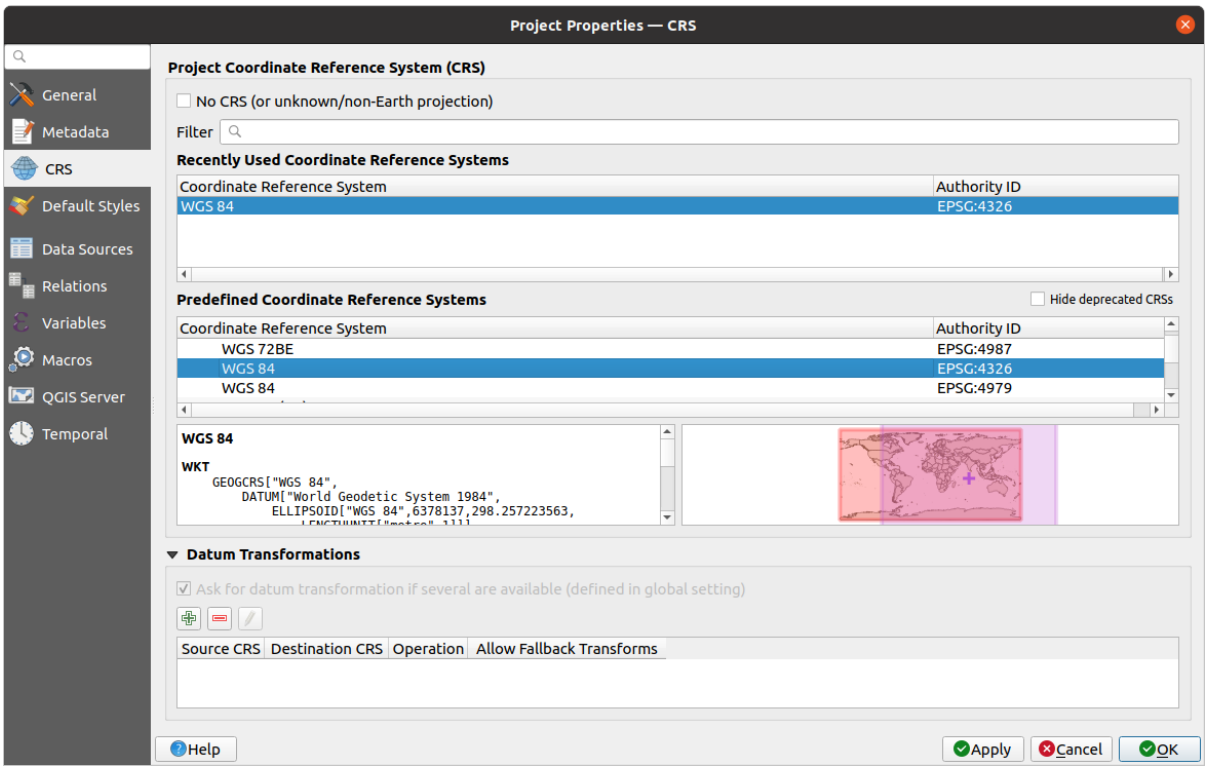


図 10.2: プロジェクトのプロパティダイアログ

利用可能なオプションは次のとおりです。

- *CRS* **なし（または未知/非地球）**：この設定をチェックすると、QGIS プロジェクト内の全ての投影処理が無効となり、全てのレイヤとマップ座標が単純な 2D デカルト座標として扱われ、地表上の位置とは無関係になります。この設定はレイヤの CRS を（生の座標に基づいて）推測したり、ロールプレイングゲームのマップや建物のマップ、微視的な事柄のような地球以外の用途で QGIS を使ったりする場合に使用できます。この場合、以下のような設定が適用されます。
  - レイヤのレンダリング時に再投影は行われません。地物は生の座標を使って描画されるだけです。
  - 距離面積の計算で前提となる楕円体は強制的に「None/Planimetric」で固定されます。
  - 距離や面積の単位、座標表示は強制的に「不明な単位」で固定され、距離や面積の測定値はすべて未知の地図単位となり、変換はできません。
- または、既存の座標参照系で **地理的座標系**、**投影された座標系** あるいは **ユーザ定義の座標系** を使用することができます。CRS の地球上における適用範囲のプレビューが表示されるため、適切な CRS を選択するのに役立ちます。プロジェクトに追加されたレイヤは、重ね合わせるために元の CRS に関係なくオンザフライでこの CRS に変換されます。使用する単位や楕円体の設定が可能かつ意味を持ち、これに応じて距離や面積の計算を行うことができます。

QGIS プロジェクトで新しい CRS を選択すると、**プロジェクトのプロパティ** ダイアログ（**プロジェクト** **プロパティ...**）の **一般情報** タブにある計測単位が選択した CRS に適合するように自動的に変更されます。例えば、一部の CRS は座標をメートルではなくフィートで定義するため、QGIS プロジェクトをこれらの CRS のいずれかに設定すると、デフォルトでフィートを使用して測定するようにプロジェクトが設定されます。

Tip: レイヤの CRS をプロジェクトに設定する

レイヤの CRS を使用して、プロジェクトに CRS を割り当てることができます。

1. **レイヤ** パネルで CRS を使用したいレイヤを右クリックする
2. **レイヤの *CRS* をプロジェクトの *CRS* に設定** を選択する

プロジェクトの CRS はレイヤの CRS で再定義されます。マップキャンバスの範囲や座標表示はこれに応じて更新され、プロジェクト内の全てのレイヤがオンザフライで新しいプロジェクトの CRS に変換されます。

10.4 座標参照系セレクタ

このダイアログには投影法データベースが備わっており、プロジェクトやレイヤに対して CRS を割り当てるのに役立ちます。ダイアログ内の項目には以下のものがあります：

- **フィルタ** ：利用したい CRS の EPSG コードの識別子または名前を知っている場合は、検索機能を使用して見つけることができます。EPSG コードの識別子または名前を入力して下さい。
- **最近使用した CRS** ：日常の GIS 作業でよく使う CRS があれば、このリストに表示されます。これらのいずれかひとつをクリックすると CRS を選択できます。
- **あらかじめ定義された CRS** ：これは、地理的座標系、投影された座標系およびユーザ定義の座標系など、QGIS でサポートされているすべての CRS のリストです。CRS を定義するには、該当するノードを展開してリストから CRS を選択します。現在アクティブな CRS は事前に選択されています。
- **PROJ text** ：投影変換エンジンである PROJ で使われる CRS 文字列です。この文字列は読み取り専用であり、情報提供のために表示されています。

座標参照系のセレクタには、選択した CRS が使用可能な地理的領域の大まかなプレビューも表示されます。多くの CRS は、狭い地理的領域でのみ使用するように設計されているため、設計された領域以外では使用しないようにしてください。プレビュー地図は、CRS がリストから選択されるたびに、おおよその使用可能領域をシェーディングします。さらに、このプレビュー地図には現在のメインキャンバス地図範囲の目安も表示されます。

10.5 カスタム座標参照系

QGIS で必要な座標参照系が提供されていない場合は、カスタム CRS を定義できます。CRS を定義するには **設定** メニューから **カスタム投影法**... を選択します。カスタム CRS は QGIS のユーザデータベースに格納されます。カスタム CRS に加えて、このデータベースには空間ブックマークやその他のカスタムデータも含まれています。

QGIS でカスタム CRS を定義するには、PROJ 投影ライブラリの十分な理解が必要です。まずは Gerald I. Evenden による 1990 年の米国地質調査所オープンファイルレポート 90-284「Cartographic Projection Procedures for the UNIX Environment - A User's Manual」を参照してください（ https://pubs.usgs.gov/of/1990/of90-284/ofr90-284.pdf から入手可能です）。

このマニュアルには、 proj および関連するコマンドラインユーティリティの使用方法が説明されています。 proj で使用される地図パラメータはこのユーザマニュアルに記載されており、QGIS も同じものを使っています。

カスタム座標参照系の定義 ダイアログでは、CRS を定義するために必要とするパラメーターは 2 つだけです。

1. わかりやすい名前
2. PROJ または WKT 形式の地図パラメータ

新しい CRS を作成するには：

1.  新規 CRS を追加 ボタンをクリックします
2. わかりやすい名前を入力します
3. 形式を選択します： *Proj* **文字列** もしくは *WKT* です
4. CRS の **パラメータ** を追加します

注釈: CRS 定義は WKT 形式による保存を推奨

Proj 文字列 と WKT の両形式をサポートしていますが、投影法の定義は WKT 形式で保存することを強くお勧めします。使用したい投影法の定義が proj 形式の場合には、まず proj 形式を選択してパラメータを入力し、それからフォーマットを WKT に切り替えます。すると、QGIS がその定義を WKT 形式に変換するので、それから保存してください。

---

5. **検証** ボタンをクリックすると、CRS の定義が問題ない投影定義であるかどうかをテストします。

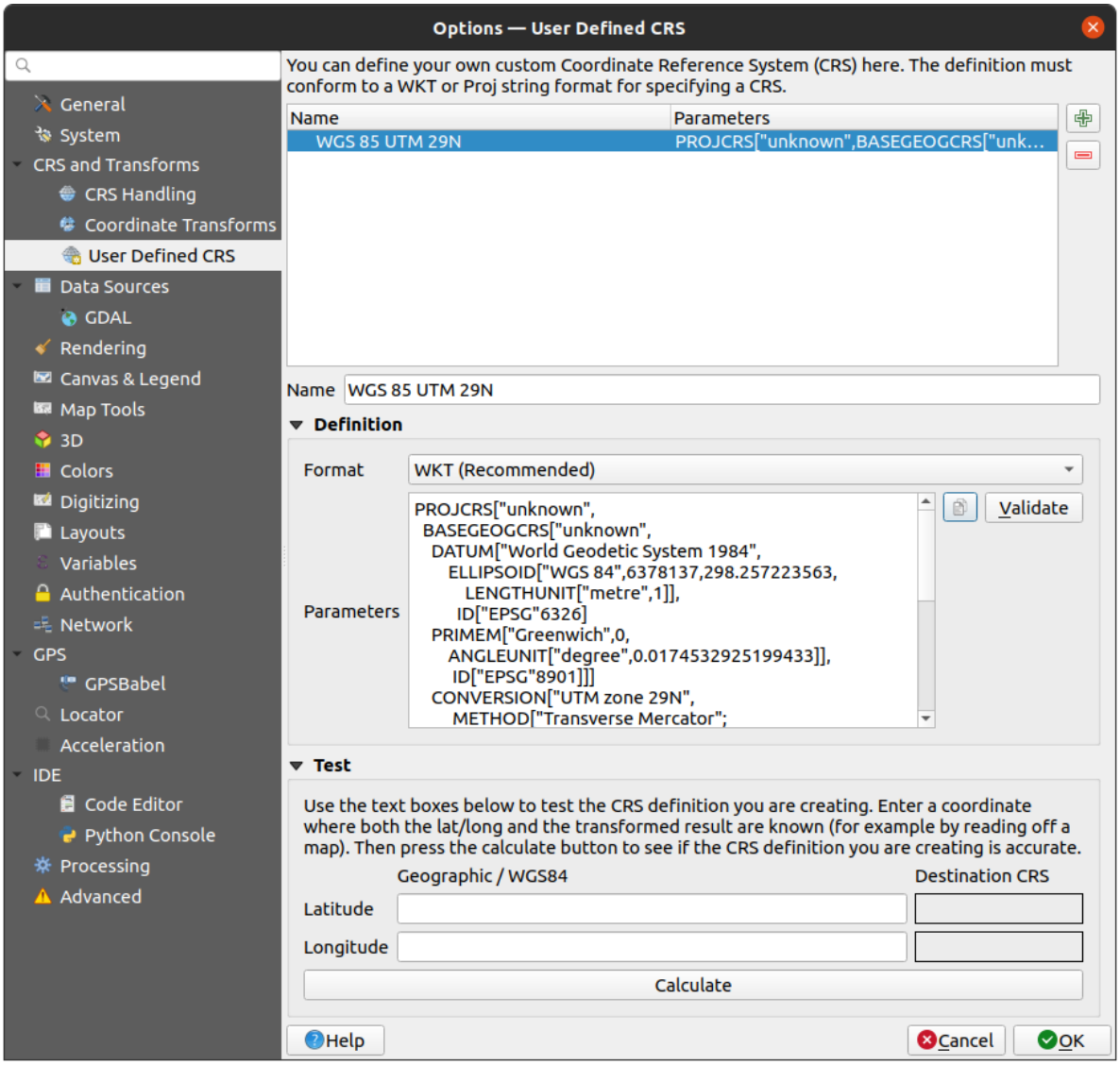


図 10.3: カスタム CRS ダイアログ

CRS のパラメータをテストして、正常な結果が得られるかどうかを確認することができます。これを行うには、既知の WGS84 の緯度と経度の値をそれぞれ **北** 、 **東** フィールドに入力します。 **計算** をクリックして、結果を座標参照系の既知の値と比較します。

10.5.1 NTv2 変換を QGIS に統合する

NTv2 変換ファイルを QGIS に統合するには、もう 1 ステップ必要です。

1. NTv2 ファイル （.gsb）を QGIS が使用する CRS/Proj フォルダに配置します（例えば windows ユーザは `C:\OSGeo4W64\share\proj` です）
2. **nadgrids** （ `+nadgrids=nameofthefile.gsb` ）を **カスタム座標系の定義** （ **設定　カスタム投影法...** ）の **パラメータ** フィールドの Proj 定義に追加します。

図 10.4: NTv2 変換の設定

10.6 測地系変換

QGIS では、デフォルトで「オンザフライ」の CRS 変換が有効になっており、異なる座標系を持つレイヤを使用すると、QGIS が透過的にプロジェクトの CRS へ再投影を行います。いくつかの CRS については、プロジェクトの CRS に再投影するために利用可能な変換が複数あります！

デフォルトでは、QGIS は利用可能な最も正確な変換を使用しようとします。しかし、変換するために追加のサポートファイルが必要な場合など、場合によってはこれが不可能な場合もあります。より正確な変換があるものの現在使用できない場合には、QGIS は警告メッセージを表示し、より正確な変換が利用可能であることと、システム上でそれを有効にするための方法を知らせてくれます。これは大抵、変換サポートファイルの外部パッケージをダウンロードし、QGIS の **ユーザープロファイル** フォルダの下にある `proj` フォルダに展開する必要があるというものです。

必要に応じて、2 つの CRS 間で複数の変換が可能な場合には QGIS はプロンプトを表示し、データに使用する最も適切な変換を情報に基づいて選択することができます。

この設定は、 **設定** 🔧 **オプション** **座標変換** タブメニューの **デフォルトの測地系変換** グループで行えます。

- **測地系変換が複数利用可能な場合は尋ねる**: をチェックすると、変換元 CRS/変換先 CRS の組み合わせに対して 2 つ以上の適切な測地系変換が存在する場合には、ダイアログが自動的に開き、プロジェクトで使用するものとしてそれらの測地系変換のうちどれを選択するかをユーザに尋ねます。このダイアログから変換を選択する際に **デフォルト変換** のチェックボックスをオンにすると、選択した内容が保存され、新しく作成された QGIS プロジェクトに自動的に適用されます。
- あるいは、プロジェクトにレイヤをロードしたり、レイヤを再投影したりする際にデフォルトとして使用する適切な測地系変換のリストを定義します。

  ボタンを押して **測地系変換を選択** ダイアログを開きます。それから、以下の手順で測地系変換リストの定義を行います。

  1. ドロップダウンメニューを使用するか、 CRS の選択 ウィジェットを使用してレイヤの **変換元** *CRS* を選択します。
  2. **変換先** *CRS* も同様に設定します。
  3. 変換元から変換先への利用可能な変換方法がテーブルに一覧表示されます。行をクリックすると、適用された設定の詳細と、対応する変換精度および変換の使用エリアが表示されます。

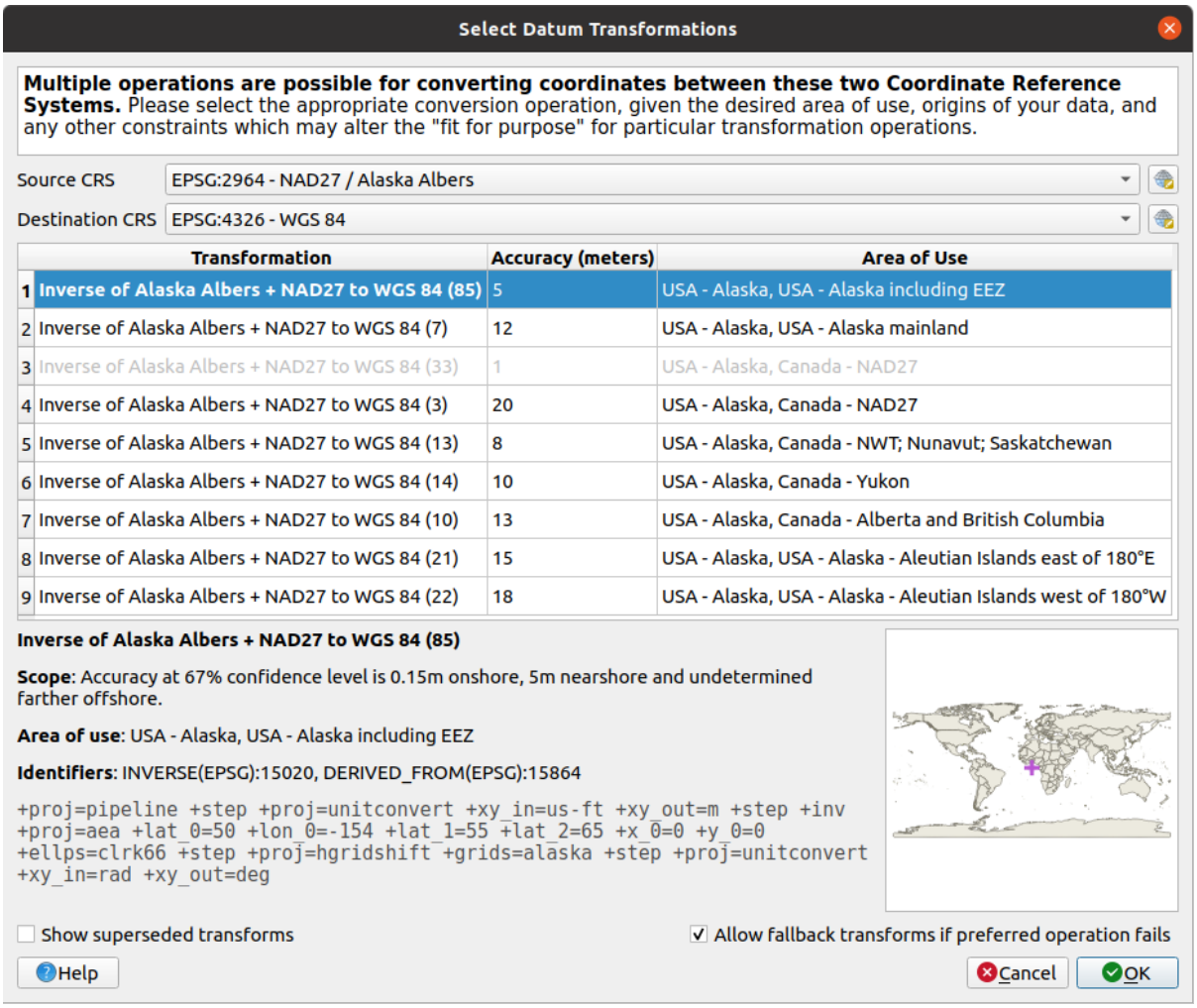


図 10.5: デフォルトの測地系変換の選択

場合によっては、ご利用のシステムでは利用できない変換があります。この場合には、変換はリストに（灰色網掛けで）表示されますが、変換サポートに必要なパッケージをダウンロードしてインストールしない限りは選択はできません。通常は、対応するグリッドをダウンロードしインストールするためのボタンがあります。このグリッドは、アクティブな **ユーザープロファイル** ディレクトリ内の `proj` フォルダに保存されます。

4. お好みの変換方法を見つけて、選択します
5. **優先される演算が失敗した場合に** *Fallback* **変換をする** かどうかを設定します
6. *OK* をクリックします。

   デフォルトの測地系変換 のテーブルに行が追加され、**変換元** *CRS* 、**変換先** *CRS* に関する情報、変換に適用される **操作** 、 *Fallback* **変換を許可** が有効になっているかどうかの情報があります。

今後は、リストからそれを削除する ( ) か、リスト内のエントリを変更する ( ) までは、QGIS は選択された測地系変換を自動的に使用して、これら 2 つの CRS 間で変換を行います。

設定 **オプション** **座標変換** タブで設定された測地系変換は、システム上で作成されるすべての新しい QGIS プロジェクトにも継承されます。さらに、プロジェクトは **プロジェクトのプロパティ** ダイアログ（ **プロジェクト** **プロパティ...** ）の **座標参照系** タブで指定した固有の変換セットを持つこともできます。

この設定は現在のプロジェクトにのみ適用されます。

第11章　地図の可視化

地図の作成は QGIS の「ビジネスエンド」の 1 つです。マップビューは、プロジェクトに読み込まれた空間レイヤを様々なレンダリング設定を適用して視覚化するためのキャンバスです。マップビューには *2D* 、*3D* 、 **標高断面図** があり、異なる縮尺や範囲での表示させたり、 **地図テーマ** を利用して読み込んだレイヤをさまざまな組み合わせで表示することができます。

11.1 2D マップビュー

2D マップビュー（ **マップキャンバス** とも呼びます）は、地図が表示される中心的な場所です。デフォルトでは、QGIS は単一のマップビュー（ メインマップ と呼ばれます）を開き、レイヤを 2D で表示し、 **レイヤ** パネルと緊密に結合しています。このウィンドウは、読み込んだレイヤに適用したレンダリング（シンボロジ、ラベル、可視性など）を反映します。

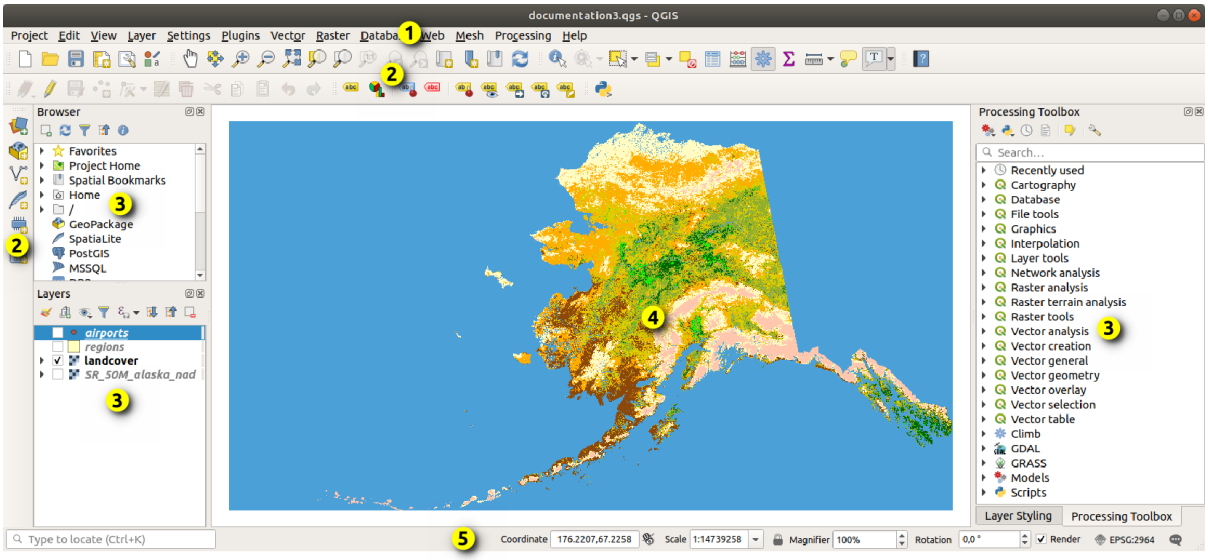


図 11.1: QGIS のユーザインタフェース

11.1.1 マップビューについて詳しくみる

レイヤを追加（たとえば **データを開く** を参照）すると、QGIS は自動的にそのレイヤの CRS を探します。空の QGIS プロジェクトから始める場合、デフォルトで別の CRS がプロジェクトに設定されているとき（**プロジェクトの座標参照系** を参照）には、レイヤの範囲は「オンザフライ」にその CRS に変換され、マップビューはその範囲にズームします。プロジェクトにすでにレイヤがある場合には、マップキャンバスのサイズ変更は実行されず、現在のマップキャンバス範囲にある地物が表示されるだけです。

マップビューをクリックして、地図のさまざまな場所にパンやズームすることでマップを操作することができます。 **ナビゲーションツールバー** と **ビュー** メニューには専用のツールがあり、キーボードやマウスボタンによるショートカットもあります。

表 11.1: マップキャンバスのナビゲーションツール

ツール	利用方法
地図を移動	• 左シングルクリック：地図はクリックした点を中心に、同じ縮尺で表示されます。 • 左マウスボタンを押しながらドラッグすると、マップキャンバスが移動します。
拡大	• 左シングルクリック：地図はクリックした点を中心に、縮尺が 2 倍で表示されます。 • 左マウスボタンでマップキャンバス上で四角形をドラッグすると、そのエリアにズームします。 • `Alt` キーを押しながらだと、縮小 ツールに切り替わります。
縮小	• 左シングルクリック：地図はクリックした点を中心に、縮尺が半分で表示されます。 • 左マウスボタンでマップキャンバス上で四角形をドラッグすると、そのエリアからズームアウトします。 • `Alt` キーを押しながらだと、拡大 ツールに切り替わります。
選択部分にパン	**レイヤ** パネルで選択されたレイヤの選択地物にマップをパンします。
選択部分にズーム	**レイヤ** パネルで選択されたレイヤの選択地物にズームします。 **レイヤのコンテキストメニューからも利用できます**
レイヤの領域にズーム	**レイヤ** パネルで選択されたレイヤすべてを合わせた範囲にズームします。 **レイヤのコンテキストメニューからも利用できます**
全域表示	プロジェクト内の全てのレイヤを含む範囲、または **プロジェクト範囲の全域** にズームします。
前の領域へズーム	マップを履歴内の前の表示範囲にズームします。

次のページに続く

表 11.1 – 前のページからの続き

ツール	利用方法
 次の表示領域にズーム	マップを履歴内の次の表示範囲にズームします。
 ネイティブ解像度にズーム（1	アクティブなラスタレイヤの 1 ピクセルがスクリーンの 1 ピクセルに一致するようにマップを拡大縮小します。 **レイヤのコンテキストメニューからも利用できます**
マウスホイール	• 地図の移動：マウスホイールを押しながらドラッグします。 • ズーム： マウスホイールを回転させると拡大・縮小します。 Ctrl キーを押しながらマウスホイールを回転させると、細かくズームできます。 • 戻る・進むボタンを押すと、マップキャンバスのズーム履歴をブラウズします。
キーボード	• 地図の移動： Space キーを押しながらマウスを移動します。矢印キーで上下左右に移動できます。 • 拡大： PgUp または Ctrl++ • 縮小： PgDown または Ctrl+- • 領域にズーム：何らかのマップツール（地物情報表示や計測ツールなど）がアクティブな時に、地図上で Shift キーを押しながら四角形をドラッグすると、その領域にズームします。選択ツールや編集ツールがアクティブな場合には対応していません。

地図上を右クリックすると、地図の CRS、WGS84 あるいはカスタム CRS で、クリックした点の  **座標をコピー** できます。コピーされた情報は式やスクリプト、テキストエディタやスプレッドシート等に貼り付けることができます。

11.1.2 地図のレンダリングの制御

デフォルトでは、QGIS はマップキャンバスが更新されるたびにすべての可視レイヤをレンダリングします。マップキャンバスの更新を発生させるイベントは次のとおりです。

- レイヤの可視性の変更
- 可視レイヤのシンボロジの変更
- レイヤの追加
- 地図のパン・ズーム
- QGIS ウィンドウのサイズ変更

QGIS ではいくつかの方法でレンダリング処理を制御できます。

- **グローバルレベル** での制御

- 例えば **縮尺に応じたレンダリング** を使用した、レイヤ毎の制御
- GUI 上の専用ツールによる制御

マップの描画を停止するには、Esc キーを押します。これによりマップキャンバスの更新は中断され、マップは部分的に描画されたままになります。ただし、 Esc を押してからマップの描画が停止するまでには少し時間がかかる場合があります。

レンダリングを中断するには、ステータスバーの右下角にある **レンダ** チェックボックスをクリックします。 **レンダ** がチェックされていない場合、QGIS は上で述べた通常のトリガーでは反応せず、キャンバスの再描画を行いません。レンダリングを中断したい場合の例としては、以下のようなものがあります：

- 多数のレイヤを追加し、描画される前にシンボルを設定したい場合
- 1つもしくは多数の巨大なレイヤを追加し、描画される前に縮尺依存表示設定を行いたい場合
- 1つもしくは多数の巨大なレイヤを追加し、描画される前に特定のビューにズームしたい場合
- 上で挙げたいずれかの組み合わせの場合

レンダ がチェックされるとレンダリングが有効になり、即座にマップキャンバスが更新されます。

11.1.3 マップキャンバスの時間制御

QGIS は読み込んだレイヤの時間制御を扱う、つまり、時間の変化に基づいてマップキャンバスのレンダリングを変更することができます。これを実現するには、以下の設定が必要です。

1. 動的な時系列プロパティが設定されているレイヤ。QGIS はカスタム設定により、データプロバイダ間で別々の時系列コントロールをサポートしています。これは主に、レイヤが表示される時間範囲を設定します。

   - **ラスタレイヤ** ：レイヤの表示・非表示をコントロールします
   - *WMTS* **レイヤ** ：固定の時間範囲に基づくか、動的な時間範囲に従って、データをレンダリングするかどうかを制御します
   - **ベクタレイヤ** ：地物は、その属性に関連付けられた時間の値に基づいてフィルタリングされます
   - **メッシュレイヤ** ：アクティブなデータセットグループの値を動的に表示します

   レイヤの動的時系列オプションを有効化した場合には、 **レイヤ** パネルのレイヤ名の隣に アイコンが表示され、そのレイヤが時系列コントロールされていることを知らせます。このアイコンをクリックすると、時系列設定を更新できます。

2. **時系列コントローラパネル** を使用して、マップキャンバスの時系列ナビゲーションを有効化します。このパネルは、以下の方法で開くことができます。

   - **ナビゲーション** ツールバー内にある 時系列コントローラパネル アイコンを使用する
   - **ビュー** **パネル** **時系列コントローラパネル** メニューを選択する

時系列コントローラパネル

時系列コントローラ パネルには、以下のモードがあります：

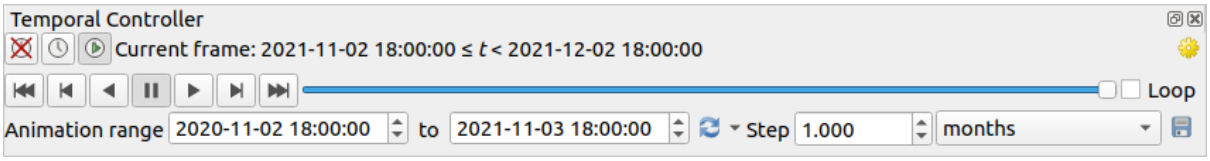


図 11.2: ナビゲーションモードの時系列コントローラパネル

- 時系列ナビを無効化 ：全ての時系列設定を無効化し、表示状態のレイヤは通常どおりにレンダリングされます
- 固定範囲の時系列ナビ ：時間範囲を設定し、レイヤ（または地物）の時間範囲がこの範囲内に重なるもののみがマップ上に表示されます
- アニメーション時系列ナビ ：時間範囲を設定し、ステップに分割して、各フレームとレイヤ（または地物）の時間範囲が重なるもののみがマップ上に表示されます
- 時系列設定 は、アニメーション全般の制御に関する設定です
  - **フレーム数/秒** ：1 秒あたりに表示されるステップ数
  - **累積範囲** ：全てのアニメーションフレームで範囲の終了日時は異なりますが、開始日時は同じになります。これは、データの時間範囲にわたって「移動する時間窓」を表示するのではなく、時系列ビジュアライゼーションでデータを累積して表示したい場合に便利です。

時系列ナビのアニメーション

アニメーションは、時間範囲内の特定の時間で変化する可視レイヤのセットに基づいています。時系列アニメーションを作成するには、以下のようにします：

1. アニメーション時系列ナビ をオンに切り替え、アニメーションプレーヤーウィジェットを表示します
2. 作成したい **アニメーション範囲** を入力します。 ボタンを使用すると、範囲を以下のいずれかで設定できます：
   - **全範囲に設定** ：時系列コントロールが有効となっているレイヤの時間の全範囲に設定します
   - **プロジェクト範囲に設定** ： **プロジェクトのプロパティ** で定義された範囲に設定します
   - **単一レイヤ範囲に設定** ：時系列コントロールが有効となっているレイヤの一つから範囲を採用します
3. 時間範囲を分割するための時間 **ステップ** を入力します。 秒 から 世紀 まで、さまざまな単位がサポートされています。 ソースのタイムスタンプ もステップとして使用できます。これを選択した場合、プロジェクト内のレイヤで利用可能な全ての時間範囲をステップした時系列ナビゲーションとなります。これは、例えば不定期な日付の画像を提供する WMS-T サービスのように、利用可能な日時が不連続となっているレイヤがプロジェクトに含まれている場合に便利です。このオプションでは、次に利用可能な画像が表示されるまでの時間間隔の分だけステップします。

4. ▶ ボタンをクリックすると、アニメーションをプレビューします。QGIS は、設定した時刻におけるレイヤのレンダリングを使用したシーンを生成します。レイヤの表示は、個々の時間フレームが時間範囲に重なるかどうかに依存します。

   アニメーションは、時間スライダーを動かすことでもプレビューできます。**ループ** チェックボックスにチェックを入れると、▶ ボタンをクリックしてアニメーションを停止するまでは、アニメーションを繰り返し再生します。ビデオプレイヤーのボタンは全て使用できます。

   （マウスがサポートしている場合には）マップキャンバス上にカーソルを置いてマウスホイールを水平スクロールすることでも時間操作ができます。「スクラブ」、つまり時系列ナビゲーションスライダーを左右に移動させることでも操作できます。

5. シーンを表す一連の画像を生成したい場合には、アニメーションを出力 ボタンをクリックします。出力画像は、後でビデオ編集ソフトで結合できます。

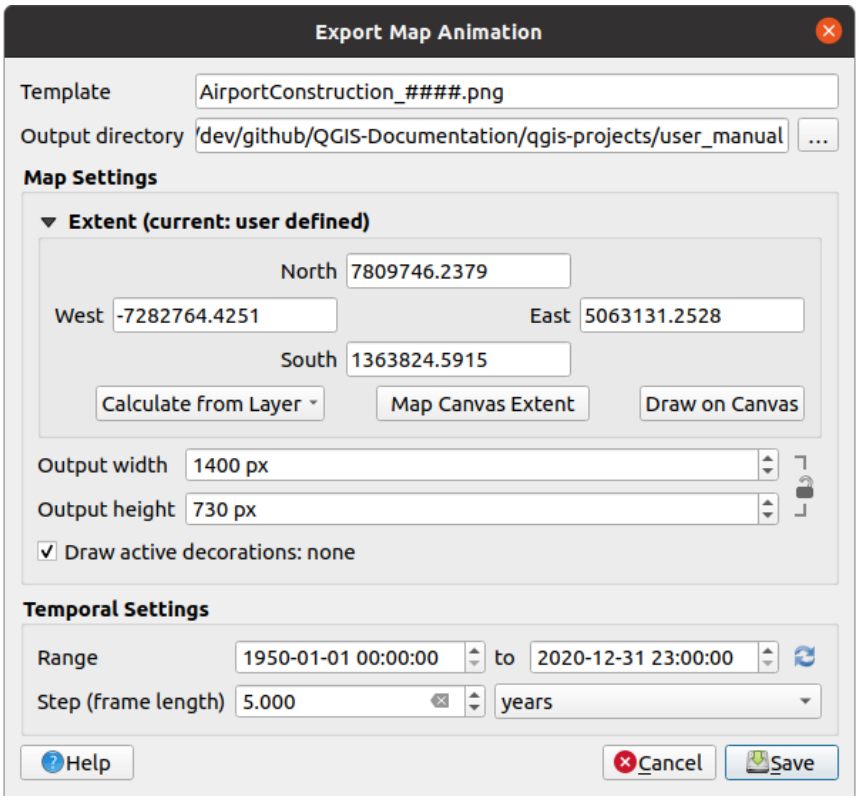


図 11.3: マップキャンバスのアニメーションシーンの画像出力

- ファイル名の **テンプレート** ： #### は、フレームの連番に置き換えられます
- **出力フォルダ**
- **地図の設定** では、以下の設定ができます：
  - 使用する **領域** の再定義
  - 画像の **解像度** の制御（ **出力幅** と **出力高** ）
  - **アクティブな装飾を描く** ：アクティブな **地図整飾** を出力にも残すかどうかを選択します
- **時系列設定** では、以下の再設定ができます：

- アニメーションの時間の **範囲**（*Range*）
- 好きな単位での **ステップ（フレーム長）**

11.1.4 地図上の範囲のブックマーク

空間ブックマークは地理的な場所を「ブックマーク」し、後でその場所に戻ることができます。デフォルトでは、ブックマークは（**ユーザー・ブックマーク** として）ユーザープロファイルに保存されるため、ユーザーが開いたどのプロジェクトからでもブックマークを利用できます。また、単一のプロジェクト に対して保存（**プロジェクト・ブックマーク**）することもでき、これはプロジェクトを他のユーザーと共有する場合に便利です。

ブックマークを作成する

ブックマークを作成するには、以下の手順で操作します：

1. 関心のある領域にズームやパンし移動します
2. メニューオプションの **ビュー** **新規空間ブックマーク**... を選択するか、 Ctrl+B を押す、もしくは **ブラウザ** パネル内の **空間ブックマーク** エントリを右クリックして、 **新規空間ブックマーク** を選択します。 **ブックマークエディタ** ダイアログが開きます。

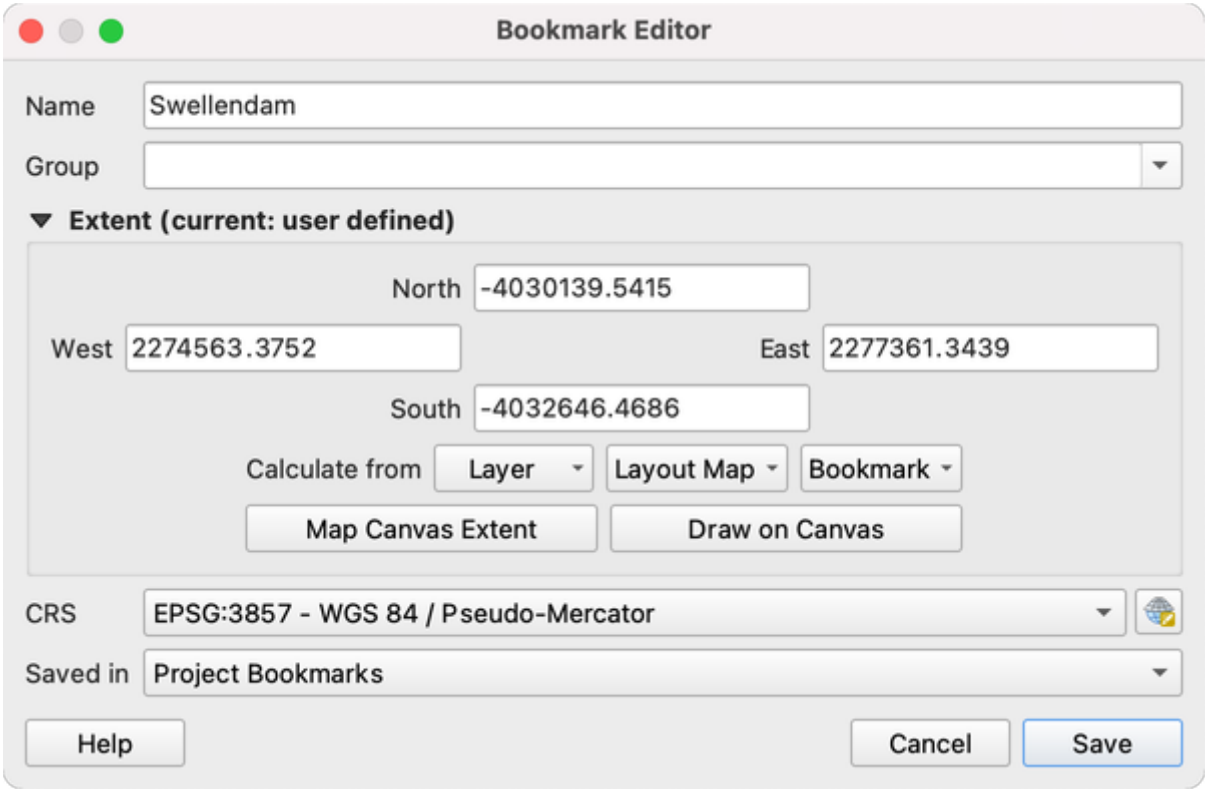


図 11.4: ブックマークエディタダイアログ

3. ブックマークにわかりやすい名前を付けます

4. 関連するブックマークを保存するグループ名を入力または選択します。
5. **範囲セレクタ** ウィジェットを使用して、保存したい領域を選択します
6. 領域に使用する *CRS* を指定します
7. ブックマークの **保存先** を **ユーザー・ブックマーク** か **プロジェクト・ブックマーク** のどちらにするか選択します（デフォルトでは、このドロップダウンリストは **ユーザー・ブックマーク** に設定されています）
8. **保存** ボタンを押して、ブックマークをリストに追加します

同じ名前のブックマークを複数持てることに注意してください。

ブックマークを操作する

ブックマークを利用・管理するには、 **空間ブックマーク** パネル もしくは **ブラウザ** パネルを使用できます。

ビュー **空間ブックマーク・マネージャを表示** を選択するか Ctrl+7 を押すと、 **空間ブックマークマネージャ** パネルが開きます。 **ビュー** **空間ブックマークを表示** を選択するか Ctrl+Shift+B を押すと、 **ブラウザ** パネル内の **空間ブックマーク** エントリが表示されます。

以下の作業を行うことができます：

表 11.2: ブックマークのアクションの管理

タスク	空間ブックマークマネージャ	ブラウザ
ブックマークにズーム	ブックマーク上でダブルクリックするか、ブックマークを選択して **ブックマークにズーム** ボタンを押す	ブックマーク上でダブルクリックするか、ブックマークをマップキャンバス上へドラッグ＆ドロップする、あるいはブックマークで右クリックし、 **ブックマークにズーム** を選択する
ブックマークを削除する	ブックマークを選択して **ブックマークを削除する** ボタンを押し、削除の確認を承認する	ブックマークを右クリックして **ブックマークを削除** を選択し、削除の確認を承認する
ブックマークをXMLへエクスポート	**ブックマークのインポートとエクスポート** ボタンをクリックし、 **エクスポート** を選択する。すべての空間ブックマーク（ユーザー、プロジェクトともに）が XML ファイルに保存される	1つまたは（ユーザー、プロジェクト）両方のフォルダ、あるいはサブフォルダ（グループ）を選択し、右クリックして **空間ブックマークをエクスポート...** を選ぶ。選択したブックマークのサブセットが保存される

次のページに続く

表 11.2 – 前のページからの続き

タスク	空間ブックマークマネージャ	ブラウザ
ブックマークを XML からインポート	**ブックマークのインポートとエクスポート** ボタンをクリックし、**インポート** を選択する。XML ファイル内の全てのブックマークがユーザー・ブックマークとしてインポートされる	ブックマークをインポートしたい場所として **空間ブックマーク** エントリか、その中のフォルダ（ユーザーまたはプロジェクト）のどれかか、あるいはそのサブフォルダ（グループ）を決めて右クリックし、**空間ブックマークをインポート** を選ぶ。 **空間ブックマーク** エントリを選んだ場合、インポートしたブックマークは **ユーザー・ブックマーク** に追加される
ブックマークを編集	テーブル内の値を変えることでブックマークを変更することができる。名前、グループ名、範囲、そしてブックマークがプロジェクトに保存されるか否かを編集できる	編集したいブックマークを右クリックし、 **空間ブックマークを編集...** を選択する。 **ブックマーク・エディタ** が開き、ブックマークを最初に作成した時のように、さまざまな点について再定義できるフォルダ間（ユーザー、プロジェクト）やサブフォルダ（グループ）間でブックマークをドラッグ＆ドロップすることもできる

ロケータ バーにブックマーク名を入力して、ブックマーク範囲にズームすることもできます。

11.1.5 地図の整飾

地図整飾には、グリッド、タイトルラベル、著作権ラベル、画像、方位記号、スケールバー、そしてレイアウト範囲があります。 これらは地図要素を追加することによって地図を「整飾」するために使われます。

グリッド

グリッド はマップキャンバスに座標グリッドと座標注釈を追加します。

1. **ビュー** **地図整飾** **グリッド...** のメニューオプションを選択し、ダイアログを開きます。

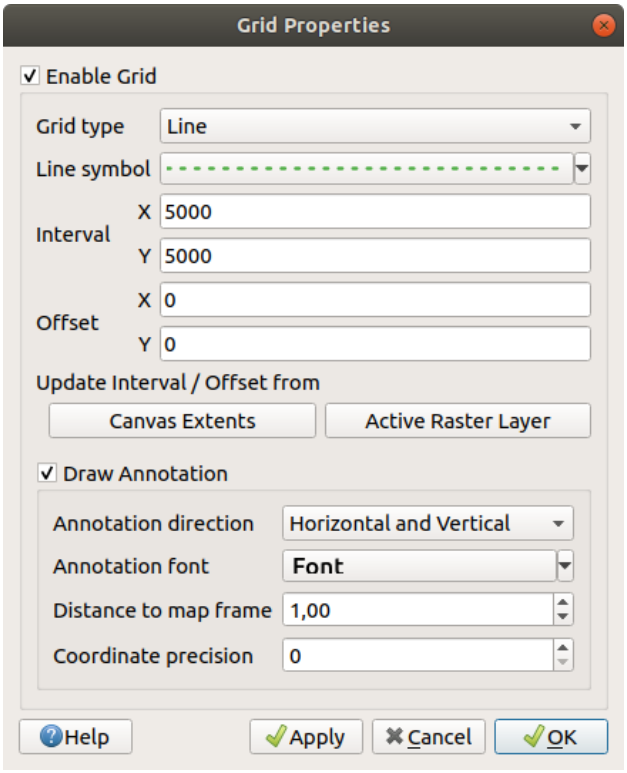


図 11.5: グリッドダイアログ

2. **グリッドを有効にする** にチェックを入れ、マップキャンバスに読み込まれたレイヤに応じてグリッド定義を設定します：
   - **グリッド型** ： **ライン** もしくは **マーカー** から選びます
   - グリッドのマークを表現するために使用される、関連した **ラインシンボル** あるいは **マーカーシンボル**
   - マップ単位によるグリッドマーク間の *X* **間隔** と *Y* **間隔**
   - マップ単位によるマップキャンバス左下隅からグリッドマークまでの距離 *X* **オフセット** と *Y* **オフセット**
   - 間隔とオフセットのパラメータは、以下に基づいて設定することもできます：
     - **キャンバスの領域**: キャンバス幅のおおよそ 1/5 の間隔でグリッドを生成します
     - **アクティブラスタレイヤ** の解像度に基づき設定します
3. **注釈の描画** にチェックを入れると、グリッドのマークの座標を表示します。以下の設定があります。
   - **注記方向** は、ラベルがグリッド線に対して相対的にどのように配置されるかを設定します。これには以下の選択肢があります：
     - 全てのラベルを **水平** あるいは **垂直** に配置する
     - **水平と垂直** 各ラベルは参照しているグリッドマークと平行です

  - **境界線の方向** 各ラベルはキャンバスの境界線に沿っており、参照するグリッドマークに垂直です

- **注記用フォント**（テキストフォーマット、バッファ、影...）**フォントセレクタウィジェット** を使って設定します。
- **地図フレームへの距離** 注記とマップキャンバスの端の間の余白距離です。 例えば画像形式や PDF に **マップキャンバスをエクスポート** する際に便利で、「紙」の境界上に注記が乗ってしまうことを避けられます。
- **座標精度**

4. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

タイトルラベル

タイトルラベル は **タイトル** でマップを整飾します。

タイトルラベル整飾を追加するには：

1. **ビュー** **地図整飾** **タイトルラベル**... のメニューオプションを選択し、ダイアログを開きます。

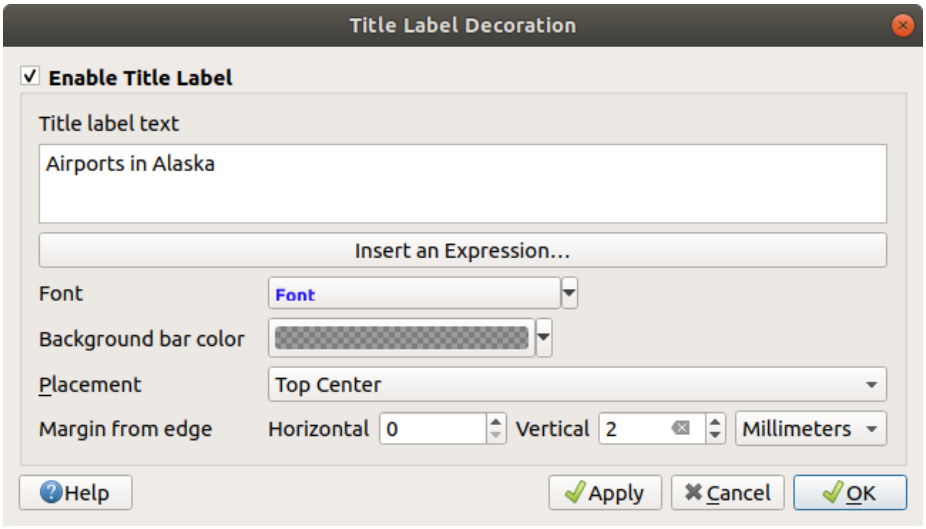


図 11.6: タイトルラベルの装飾ダイアログ

2. **タイトルラベルの有効化** をチェックします
3. マップに追加したいタイトルのテキストを入力します。 **式の挿入・編集**... ボタンを使用して、テキストを動的にすることもできます。
4. ラベルの **フォント** を QGIS の **テキストの書式設定** オプションへフルアクセスできる **フォントセレクタウィジェット** を使用して選択します。フォントコンボボックスの右にある黒い矢印をクリックすると、フォントの色や不透明度を素早く設定することができます。
5. タイトルの **背景のバーの色** に適用する **色** を選択します。

6. キャンバス内のラベルの **配置** を選択します。選択肢は **左上** 、 **中上** （デフォルト）、 **右上** 、 **左下** 、 **中下** そして **右下** です。

7. 水平および/または垂直の **端からのマージン** を設定することで、アイテムの配置を調整します。これらの値は **ミリメートル** または **ピクセル** 単位で指定するか、マップキャンバスの幅または高さの **パーセント** として設定することができます。

8. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

著作権ラベル

著作権ラベル は、 **著作権** ラベルでマップを整飾するために使います。

著作権ラベル整飾を追加するには：

1. **ビュー** **地図整飾** **著作権ラベル**... のメニューオプションを選択し、ダイアログを開きます。

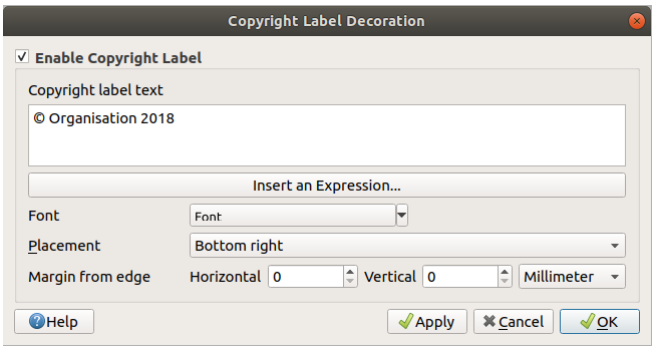


図 11.7: 著作権ラベル整飾ダイアログ

2. **著作権ラベルを有効にする** をチェックしてください

3. マップに追加したい著作権ラベルのテキストを入力します。 **式の挿入・編集**... ボタンを使用して、テキストを動的にすることもできます。

4. ラベルの **フォント** を QGIS の **テキストの書式設定** オプションへフルアクセスできる **フォントセレクタウィジェット** を使用して選択します。フォントコンボボックスの右にある黒い矢印をクリックすると、フォントの色や不透明度を素早く設定することができます。

5. キャンバス内のラベルの **配置** を選択します。選択肢は **左上** 、 **中上** 、 **右上** 、 **左下** 、 **中下** そして **右下（著作権整飾のデフォルト）** です。

6. 水平および/または垂直の **端からのマージン** を設定することで、アイテムの配置を調整します。これらの値は **ミリメートル** または **ピクセル** 単位で指定するか、マップキャンバスの幅または高さの **パーセント** として設定することができます。

7. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

画像

画像 はマップキャンバスに画像（ロゴ、凡例など...）を追加します。

画像を追加するには：

1. **ビュー** **地図整飾** **画像**... のメニューオプションを選択し、ダイアログを開きます。

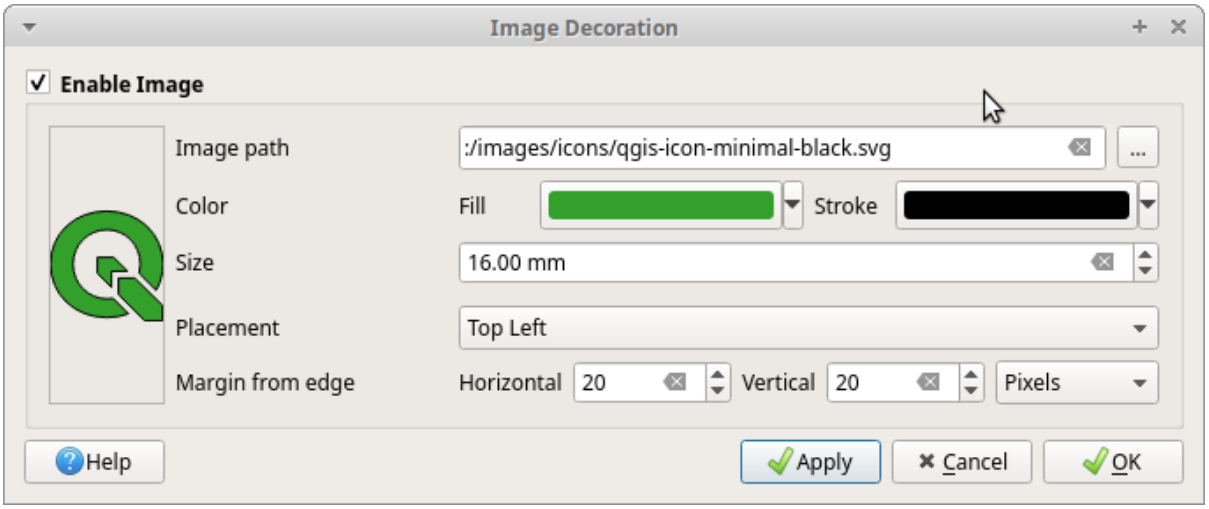


図 11.8: 画像整飾ダイアログ

2. **画像を有効化** をチェックしてください
3. ... ブラウズ ボタンを押して、ビットマップ画像（例：png や jpg）か SVG 画像を選択します
4. パラメータが有効化された SVG 画像を選択した場合、 **塗りつぶし** や **ストローク** （輪郭線）の色を設定することもできます。ビットマップ画像の場合には、この色設定は無効になっています。
5. 画像の **大きさ** をミリメートル単位で指定します。選択された画像の幅は、与えられた **大きさ** にリサイズされます。
6. **配置** コンボボックスを用いて、マップ上で画像を配置したい場所を選択します。 デフォルトの位置は **左上** です。
7. **（キャンバス）端からの水平** および **垂直 マージン** を設定します。これらの値は **ミリメートル** または **ピクセル** 単位で指定するか、マップキャンバスの幅または高さの **パーセント** として設定することができます。
8. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

方位記号

方位記号 はマップキャンバスに北向き矢印を追加します。

方位記号を追加するには：

1. **ビュー** **地図整飾** **方位記号**... のメニューオプションを選択し、ダイアログを開きます。

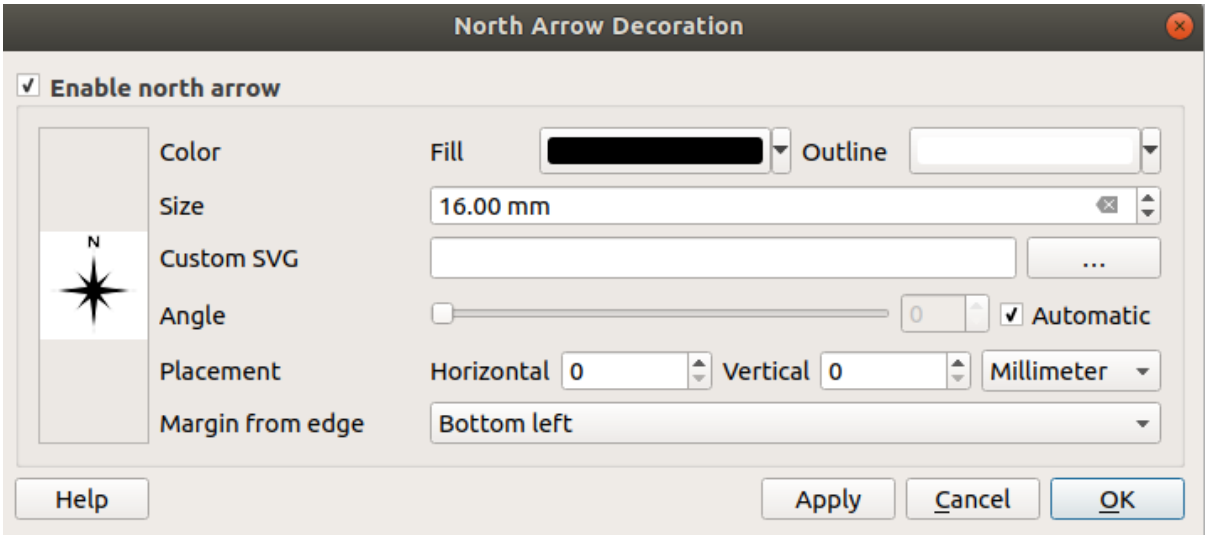


図 11.9: 方位記号ダイアログ

2. **方位記号を使用** をチェックしてください
3. オプションで色やサイズを変更したり、カスタム SVG を選択したりすることができます。
4. オプションで角度を変更するか、あるいは QGIS に **自動** で方向を決定させることができます。
5. オプションで「配置」コンボボックスから配置位置を選択できます。
6. 必要に応じて、水平および/または垂直の **（キャンバス）端からのマージン** を設定して矢印の配置を調整します。これらの値は **ミリメートル** または **ピクセル** 単位で指定するか、マップキャンバスの幅または高さの **パーセント** として設定できます。
7. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

スケールバー

スケールバー は、マップキャンバスに単純なスケールバーを追加します。スタイルや配置、バーのラベルを制御することができます。

QGIS はマップフレームと同じ単位でのスケール表示しかサポートしていません。従って、プロジェクトの CRS の単位がメートルの場合には、フィート単位のスケールバーを作成することはできません。同様に、小数点表示の角度を使っている場合には、距離をメートル単位で表示するスケールバーは作成できません。

スケールバーを追加するには：

1. **ビュー** **地図整飾** **スケールバー**... のメニューオプションを選択し、ダイアログを開きます。

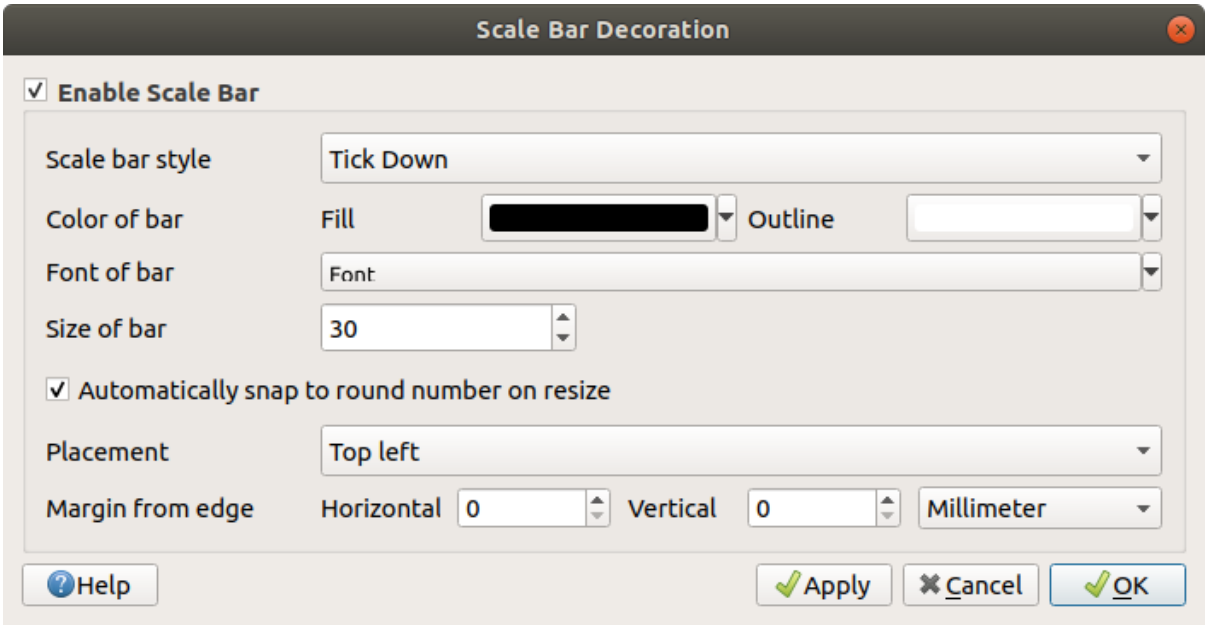


図 11.10: スケールバーダイアログ

2. **スケールバーを有効にする** をチェックしてください
3. **スケールバーのスタイル** コンボボックスからスタイルを選択します
4. 塗りつぶし色（デフォルト：黒）とアウトライン色（デフォルト：白）を選択して、 **バーの色** を設定します。色入力の右にある下矢印をクリックすると、スケールバーの塗りつぶしとアウトラインの不透明度を設定することもできます。
5. **バーのフォント** コンボボックスからスケールバーのフォントを選択します
6. **バーのサイズ** を設定します
7. オプションで **リサイズ時に自動的に四捨五入** をチェックすることで、読みやすい値で表示できます
8. **配置** コンボボックスから配置位置を選択します
9. 水平および/または垂直の **（キャンバス）端からのマージン** を設定することで、アイテムの配置を調整します。これらの値は **ミリメートル** または **ピクセル** 単位で指定するか、マップキャンバスの幅または高さの **パーセント** として設定できます。
10. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

レイアウト範囲

レイアウト範囲 は印刷レイアウトの **地図アイテム** の範囲をキャンバスに追加します。 有効にすると、すべての印刷レイアウト内のすべての地図アイテムの範囲が、印刷レイアウトと地図アイテムの名前でラベル付けされた淡い点線の境界線を使用して表示されます。表示されたレイアウト範囲のスタイルとラベル付けを制御できます。この装飾は、ラベルなどの地図要素の位置を微調整していて、印刷レイアウトの実際の表示領域を知る必要がある場合に役立ちます。

図 11.11: 2 つの印刷レイアウトのレイアウト範囲を QGIS プロジェクトに表示した例。「Sights」という名前の印刷レイアウトには 2 つの地図アイテムがあり、もう一つの印刷レイアウトの地図アイテムは 1 つ。

レイアウト範囲を追加するには：

1. **ビュー** → **地図整飾** → **レイアウト範囲**... を選択し、ダイアログを開きます。

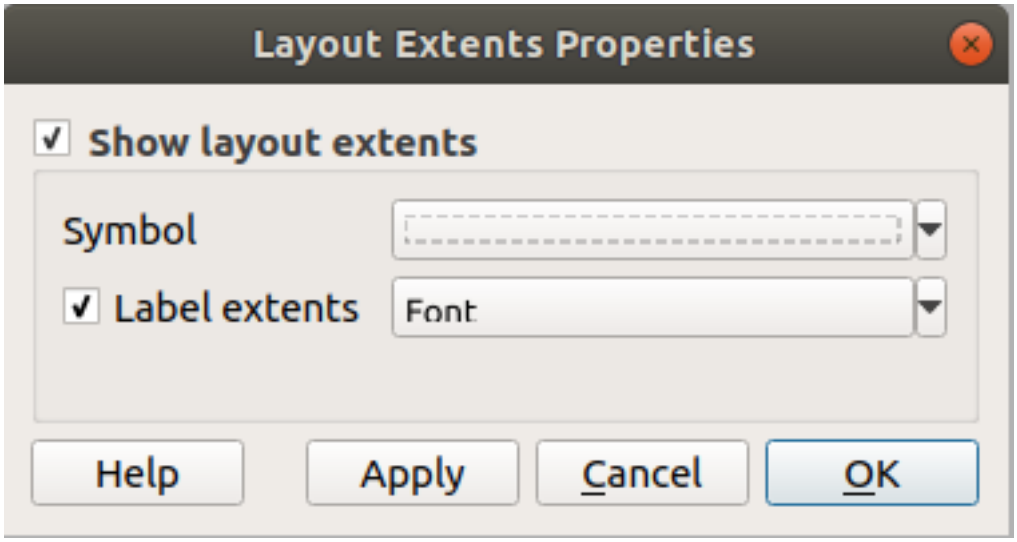


図 11.12: レイアウト範囲ダイアログ

2. **レイアウトの範囲を表示する** をチェックしてください。
3. オプションで範囲のシンボルとラベル付けを変更することができます。
4. **適用** ボタンをクリックして見た目が期待通りかを確認し、満足ならば *OK* ボタンをクリックします。

Tip: 地図整飾の設定

QGIS プロジェクトファイルを保存する際、グリッド、方位記号、スケールバー、著作権およびレイアウト範囲に加えた変更はそのプロジェクトに保存され、次回プロジェクトをロードした際に復元されます。

11.1.6 注記ツール

注記は、レンダリングされたレイヤでは表現することができない追加情報を提供するための、マップキャンバスに追加されるもう一つの種類の要素です。ベクタレイヤが持つ属性値に依存する **ラベル** とは異なり、注記は独立した詳細情報で、プロジェクト自体に格納されます。

QGIS では 2 種類の注記が利用できます:

- **地物注記**: これは、実際にジオリファレンスされたテキスト、マーカー、ライン、またはポリゴンタイプの地物で、「注記レイヤ」と呼ばれる特別なレイヤに保存されます。この注記は特定の地理的な位置に紐づいているため、地図を移動したり、縮尺や投影法を変えたりしても、注記が地図上のあちこちに行ってしまうことはありません。そうではなく、注記を描画した位置に固定されます。
- **バルーン注記**: これは、テキスト、フォーム、または画像形式の独立した注記で、吹き出し内に配置されます。この注記は、視認性を向上させる目的でどんなレイヤにも関連付けることができ、マップキャンバスの最上位に表示されます。注記のサイズはマップキャンバスの縮尺に依存し、位置は固定できます。

Tip: マップに注記をレイアウトする

以下の方法で、注記をマップとともにさまざまな形式で印刷またはエクスポートすることができます。

- **プロジェクト** メニュー内にあるマップキャンバスのエクスポートツール
- **印刷レイアウト** の場合には、対応する地図のアイテムプロパティで **地図キャンバスアイテムの描画** にチェックを入れる必要があります。

注記ツールバー には、両方の種類の注記を作成したり操作したりするためのツールがあります。

表 11.3: 注記ツールバーのアクション

ツール	利用方法	スコープ
 新規注記レイヤ	注記を保存するための新しいレイヤを作成します	地物注記
メイン注記レイヤの属	メイン注記レイヤの設定を行います	
 注記を修正	注記の選択や移動、大きさの変更や、注記のシンボロジのプロパティを修正します	
 ポリゴン注記を作成	ポリゴン地物の注記を作成します	
 ライン注記を作成	ポリライン地物の注記を作成します	
 マーカー注記を作成	ポイント地物の注記を作成します	
 点上の注記を作成	テキストラベルの注記を作成します	
 注記	テキスト形式の注記の選択と作成ができます	バルーン注記
 HTML 注記	HTML ファイルの内容の注記の選択と作成ができます	
 SVG 注記	SVG ファイルを表示する注記の選択と作成ができます	
 注記フォーム	カスタムフォームファイルの形式でベクタレイヤの属性を表示する注記の選択と作成ができます	
 注記を移動	注記要素の大きさや位置を調整します	

地物注記

地物注記は **注記レイヤ** に保存されます。通常のレイヤとは異なり、注記レイヤは現在のプロジェクトでのみ利用可能で、さまざまなタイプ（テキスト、マーカー、ライン、ポリゴン）の地物を持つことができます。このレイヤは属性テーブルを持たず、関連付けられたシンボロジもありませんが、その代わり、各地物は **レイヤスタイル** パネルを使用してアイテム毎にシンボルを設定できます。

QGIS には 2 種類の注記レイヤがあります:

- 通常の **注記レイヤ**: これは  新規注記レイヤ ツールを使用して作成できます。このレイヤは **レイヤ** パネルに表示され、一般的なレイヤと同様に、地物の可視性を制御したり、マップ内の特定のレイヤの

上または下に表示するように移動させたりすることができます。レイヤをダブルクリックすると、プロパティにアクセスできます。

- **メイン注記レイヤ**: デフォルトでは、プロジェクトに注記レイヤがない場合、または注記の作成時に注記レイヤが選択されていない場合、メイン注記レイヤに注記が保存されます。このレイヤは常にマップの一番上に描画され、プロジェクト内のその他のレイヤと並んで **レイヤ** パネルに表示されることはありません。つまり、このレイヤの地物は常に表示されます。 **注記** ツールバーの **メイン注記レイヤの属性**... エントリを使用すると、メイン注記レイヤのプロパティダイアログを開きます。

インタラクション

地物注記には、その種類に応じた専用の作成ツールがあります:

-  ポリゴン注記を作成
-  ライン注記を作成
-  マーカー注記を作成
-  点上の注記を作成

地物を作成する際の通常の QGIS ショートカットはすべて、注記アイテムを作成するときにも適用されます。ラインまたはポリゴンの注記は、各頂点を左クリックし、最後にマウスの右クリックで図形作成を終了することで描画します。描画にあたってスナップを有効にでき、 **高度なデジタイズツール** を使用して頂点を正確に配置したり、 **描画ツール** をストリーム・デジタイジングモードに切り替えて、完全に自由な形状を作成することさえもできます。

通常のレイヤとは異なり、注記レイヤは地物を選択する前にレイヤをアクティブにする必要はありません。単に  注記を修正 ツールを使用するだけで、任意の地物注記の操作ができます:

- **選択**: 注記を左クリック
- **移動**: 選択した注記アイテムを左クリックすることで移動を開始。右クリックまたは Esc キーを押すと移動をキャンセル。2 回目の左クリックで移動を確定。移動量はカーソルキーを押すことでも制御可:
  - Shift+key ：大きく移動
  - Alt+key ： 1 ピクセル だけ移動
- **ジオメトリの修正**: ライン注記やポリゴン注記の場合は、ジオメトリの頂点上を左クリックし、移動してからもう一度左クリック。セグメントをダブルクリックすると、新しい頂点を追加。
- **削除**: 注記が選択されている時に Del キーまたは Backspace キーを押すと、その注記を削除
- **地物のシンボロジの変更**

地物のシンボロジ

選択した注記は、 **レイヤスタイル** パネルに **シンボロジ** プロパティを表示します。ここでは、次の操作が可能です:

- 見た目（テキスト自体も含む）の修正を、注記の種類に応じて **シンボル** や **テキストフォーマット** の全機能を使用して行えます。
- **参照スケール** の設定
- *Z-index* の設定
- **レイヤレンダリング** の設定の修正

レイヤプロパティ

注記レイヤのプロパティダイアログには、以下のタブがあります:

- **情報**: 読み取り専用のダイアログで、現在のレイヤの要約された情報やメタデータをさっと掴むことができる興味深い場所です。
- **ソース**: 注記レイヤの一般的な設定を定義します。以下の設定ができます:
  - プロジェクト内（ **レイヤ** パネルや式など）でレイヤを識別するのに使用する **レイヤ名** を設定できます。
  - レイヤに **設定された** *CRS* を表示できます: 最近使用した CRS をドロップダウンリストから選ぶか、 CRS を選択 ボタン（ **座標参照系セレクタ** 参照）をクリックすることで、レイヤの CRS を変更できます。 レイヤの CRS が間違っている場合か、CRS が何も設定されていない場合にのみ、この操作を行ってください。
- **レンダリング**:
  - **最大縮尺（含む）** と **最小縮尺（含まない）** を設定して、地物が表示される縮尺の範囲を定義できます。この範囲の外では地物は非表示になります。 現在のキャンバスの縮尺に設定 ボタンを使用すると、可視性の範囲の境界として現在のマップキャンバスの縮尺を使用できます。詳細については **表示縮尺セレクタ** 参照してください。
  - **不透明度** ：このツールを使用すると、マップキャンバスで背面にあるレイヤを見えるようにできます。スライダーを使用して、レイヤの見え方を必要に応じて変化させてください。スライダーの横にあるメニューで不透明度の割合を正確に定義することもできます。
  - **レイヤ** レベルの **混合モード** ：このツールを使用すると、これまではグラフィックソフトでしか使えなかったような、特別なレンダリング効果が得られます。上下のレイヤのピクセルは、 **混合モード** で説明されている設定で混合されます。
  - **描画エフェクト** ボタンを使用して、レイヤの地物すべてに **描画効果** を適用します。

  これらのオプションの一部は、地物注記の **シンボロジ** プロパティからアクセスできます。

バルーン注記

バルーン注記は **編集**　**注記を追加**　メニューか、 **注記ツールバー** から追加できます:

- 文字注記 カスタム書式テキストの注記
- HTML 注記 `html` ファイルのコンテンツを配置するための注記
- SVG 注記 `SVG` ファイルのシンボルを追加するための注記
- 注記フォーム: ベクタレイヤの属性を、カスタマイズされた `ui` ファイル（図 11.13 を参照）で表示するのに便利です。 **カスタム属性フォーム** に似ていますが、これは注記アイテムに表示されます。詳細については、Tim Sutton 氏によるビデオ https://www.youtube.com/watch?v=0pDBuSbQ02o&feature=youtu.be&t=2m25s も参照してください。
- 注記を移動 （クリック＆ドラッグで）注記要素の大きさや位置を調節します

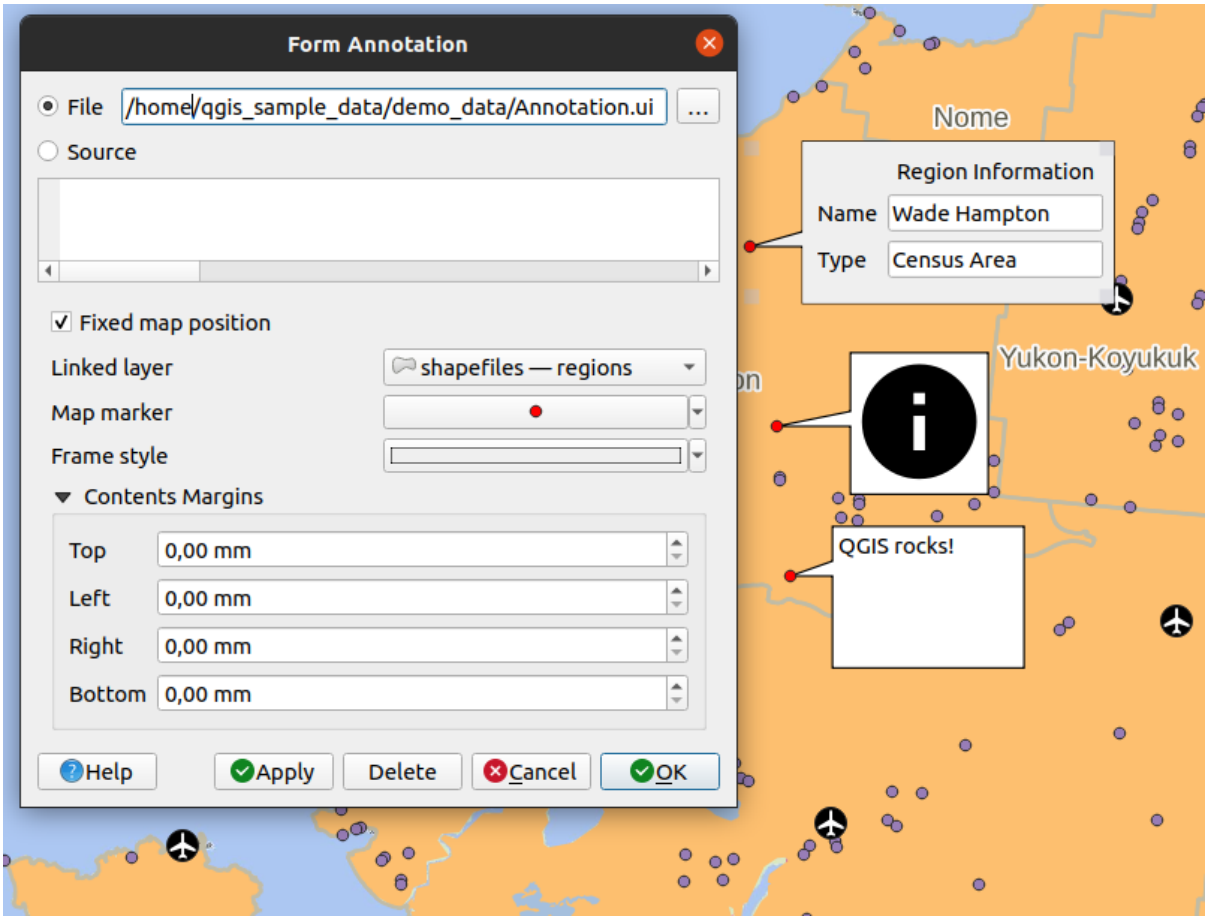


図 11.13: バルーン注記の例

バルーン注記を追加するには、対応するツールを選択してマップキャンバスをクリックします。空のバルーンが追加されます。これをダブルクリックすると、さまざまなオプションを含むダイアログが開きます。このダイアログは、すべての注記タイプでほぼ同じです。

- 一番上には、注記の種類に応じて `html` 、 `svg` または `ui` ファイルへのパスを入力するためのファイルセレクタがあります。テキスト注釈の場合は、テキストボックスにメッセージを入力して、通常の

フォントツールを使用してそのレンダリングを設定できます。

- **地図の固定位置**：チェックを外すと、バルーンの配置は（地図ではなく）画面の位置に基づきます。つまり、注記は地図のキャンバスの範囲に関係なく常に表示されます。
- **リンクされたレイヤ**: マップのレイヤに注記を関連付け、そのレイヤが表示状態のときのみ注記が表示されるようにします。
- **マーカー**: *QGIS* **のシンボル** を使って、バルーンのアンカー位置に表示されるシンボルを設定します（**地図の固定位置** にチェックが入っているときのみ表示されます）。
- **フレームスタイル**: QGIS のシンボル設定を使用して、フレームの背景色や透明度、バルーンのストローク色やストローク幅を設定します。
- **コンテンツのマージン**: 注記フレームの内側のマージンを設定します。

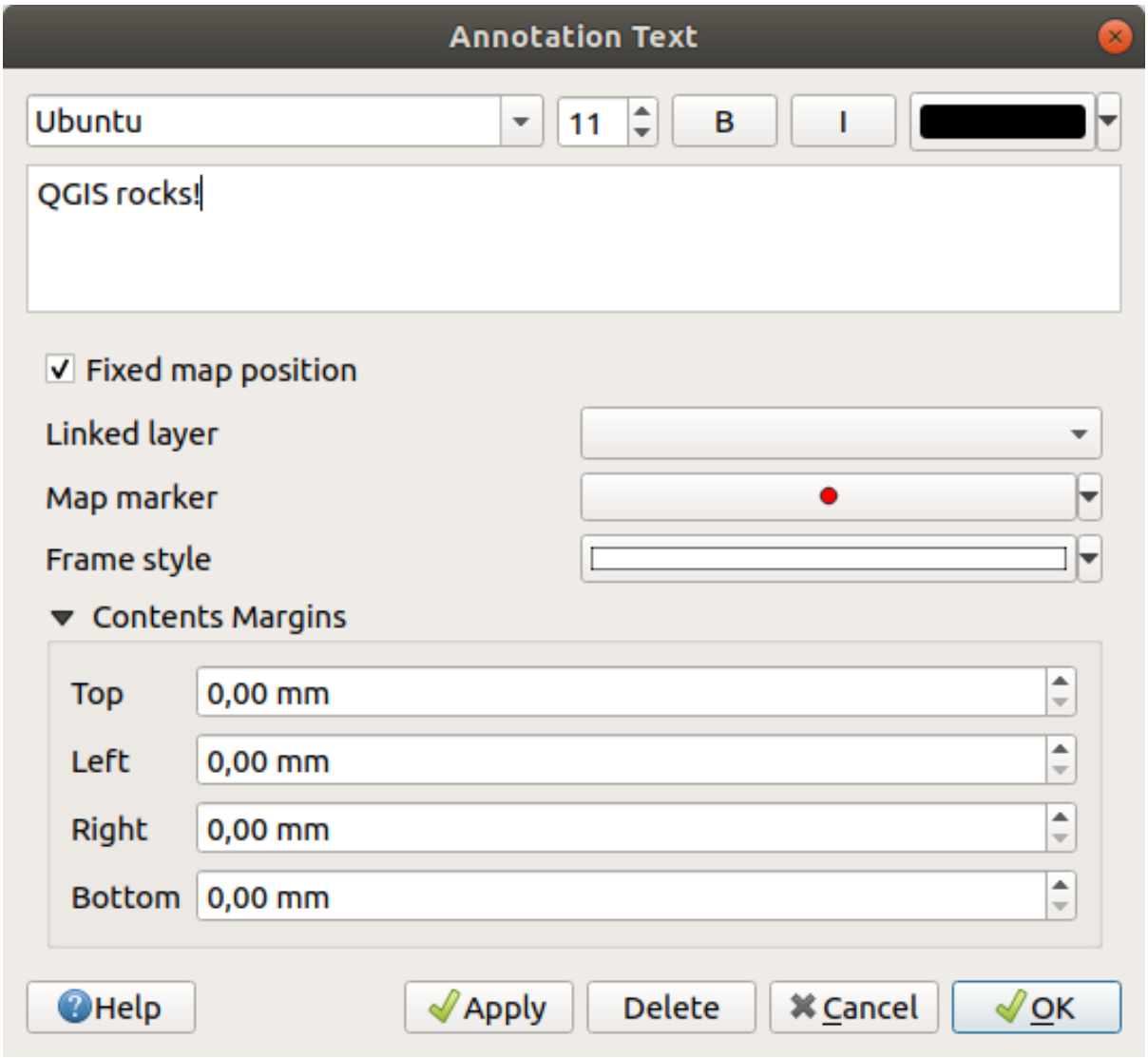


図 11.14: バルーン注記のテキストダイアログ

注記ツールが有効なときは、注記を選択できます。注記は、（マップマーカーをドラッグすることで）地図上の位置を移動したり、バルーンだけを移動したりすることができます。 注記を移動 ツールでマップ

キャンバス上のバルーンを移動させることもできます。

注記を削除するには、その注記を選択して Del または Backspace キーを押すか、またはその注記をダブルクリックしてプロパティダイアログの **削除** ボタンを押します。

注釈: バルーン **注記** ツール（注記の移動、文字注記、注記フォーム ）が有効な時に Ctrl+T を押すと、アイテムの表示・非表示が反転します。

11.1.7 計測

一般情報

QGIS は、ジオメトリを計測する 4 つの手段を提供します。

- インタラクティブな計測ツール 
-  フィールド計算機 による計測
- **地物の識別** ツールによる派生した属性の計測
- ベクタ解析ツール: **ベクタ　　ジオメトリツール　　ジオメトリ属性の追加**

計測は投影された座標系（例えば UTM）でも、非投影データでも機能します。最初の 3 つの計測ツールは、グローバルなプロジェクト設定に対しても同様に動作します：

- 他のほとんどの GIS とは異なり、QGIS のデフォルトの計測基準は楕円体で、 **プロジェクト　　プロパティ...　　一般情報** で定義される楕円体を使っています。これは、プロジェクトに地理座標系や投影座標系が定義されている場合にも当てはまります。
- 投影された/平面的な面積や距離を直交座標系によって計算したい場合には、計測の楕円体を「None/Planimetric」にしなければなりません（ **プロジェクト　　プロパティ...　　一般情報** ）。ただし、データとプロジェクトに定義された地理的（すなわち非投影の）CRS を使用すると、面積と距離の計測は楕円体計算になります。

しかしながら、地物情報表示ツールやフィールド計算機はどちらも計測前にデータをプロジェクトの CRS へ変換しません。これを行いたい場合には、ベクタ解析ツール: **ベクタ　　ジオメトリツール　　ジオメトリ属性の追加**... を使用する必要があります。ここでは、楕円体計測を選択しない限り、計測は平面座標系で行われます。

対話的に長さ、面積、方位、角度を計測

属性ツールバーの  アイコンをクリックすると、計測を開始できます。アイコンの近くにある下矢印で、  長さ、  面積、  方位、  角度の計測を切り替えられます。ダイアログで使用されるデフォルトの単位は、 **プロジェクト　プロパティ...　一般情報** メニュー内で設定された単位です。

線の長さを測る と **面積を測る** では、計測は  **デカルト座標** または  **回転楕円体** による計測で行うことができます。

注釈: 計測ツールを設定する

長さや面積を測定しているときに、ウィジェットの下にある **設定** ボタンをクリックすると、 **設定　オプション　ツール** メニューが開きます。ここではラバーバンド色、測定の桁数、計測単位を選択できます。好きな長さや角度の単位も選択できますが、現在のプロジェクト中ではこれらの値は **プロジェクト　プロパティ　一般情報** メニューで行われた選択と計測ウィジェットで行われた選択によって上書きされることに注意してください。

すべての計測モジュールは、デジタイジングモジュールのスナップ設定（ **スナップ許容範囲と検索半径の設定** セクションを参照）を使用します。正確にライン地物に沿って、あるいはポリゴン地物の周りを測定したいのであれば、最初にそのレイヤのスナップ許容誤差を設定します。すると、測定ツールを使用している場合の（許容誤差の設定内の）各マウスクリックはそのレイヤにスナップします。

 線の長さを測る は、与えられた点間の距離を測ります。このツールは、マップ上の点を複数クリックできます。各セグメントの長さとその合計が計測ウィンドウに表示されます。計測を停止するには、マウスの右ボタンをクリックします。 **すべてコピー** ボタンを使用すると、全ての線の計測結果をクリップボードに一度にコピーできます。

計測ツールでの作業中に、合計の近くにあるドロップダウンリストを操作して測定単位を対話的に変更できます、(「メートル」、「キロメートル」、「フィート」、「ヤード」、「マイル」、「海里」「センチメートル」、「ミリメートル」、「度」、「地図単位」)。この単位は、新しいプロジェクトが作成されるか、別のプロジェクトが開かれるまで、このウィジェットで保持されます。

ダイアログの **情報** セクションでは、利用可能な CRS 設定に従ってどのように計算が行われるかが説明されています。

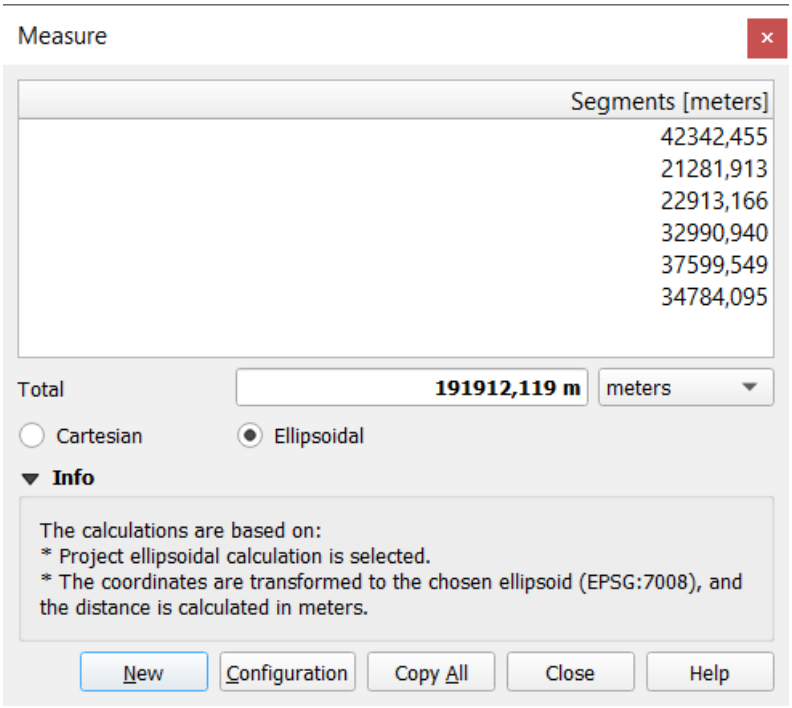


図 11.15: 距離の計測

面積を測る: 面積も計測することができます。計測ウィンドウには、累積の面積合計が表示されます。右クリックで描画を停止します。情報セクションがあり、異なる面積単位への変換機能（「平方メートル」、「平方キロメートル」、「平方フィート」、「平方ヤード」、「平方マイル」、「ヘクタール」、「エーカー」、「平方センチメートル」、「平方ミリメートル」、「平方海里」、「平方度」、「地図単位」）もあります。

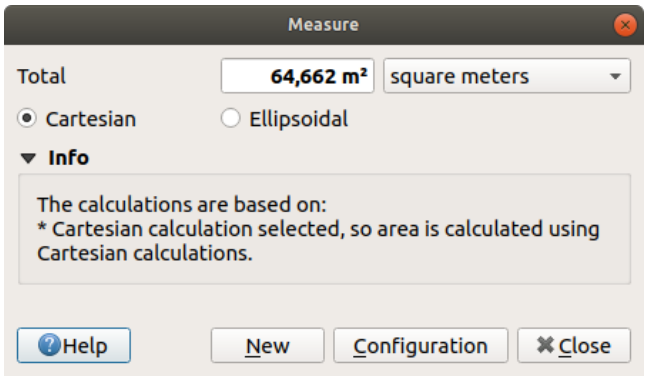


図 11.16: 面積の計測

方位を測る: 方位も計測できます。カーソルが十字型に変化します。方位の起点をクリックし、カーソルを動かして 2 点目を描きます。計測値はポップアップダイアログに表示されます。

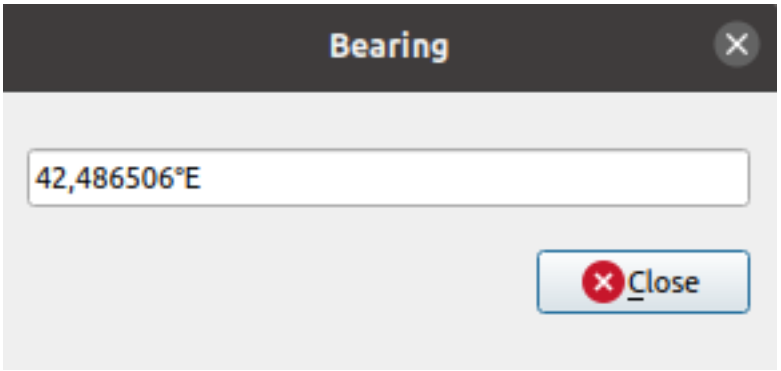


図 11.17: 方位の計測

角度を測る: 角度を測ることもできます。カーソルが十字型に変化します。クリックして計測したい角の 1 つ目のセグメントを描画し、それからカーソルを動かして求める角を描きます。計測値はポップアップダイアログに表示されます。

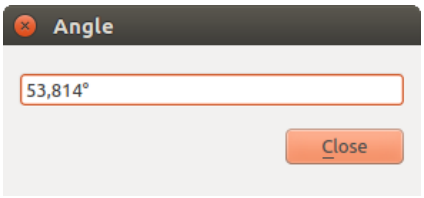


図 11.18: 角度の計測

11.1.8 追加のマップビューの設定

また、 **レイヤ** パネルの現在の状態とは違った内容のマップビューを追加で開くこともできます。新しいマップビューを追加するには、**ビュー** **新規マップビュー** を押します。すると、メインマップビューのレンダリングをコピーする新しいフローティングウィジェットが QGIS に追加されます。 マップビューは必要なだけ追加することができます。追加したマップビューはフローティングのままにすることも、縦横に並べたり、上に積み重ねたりすることもできます。

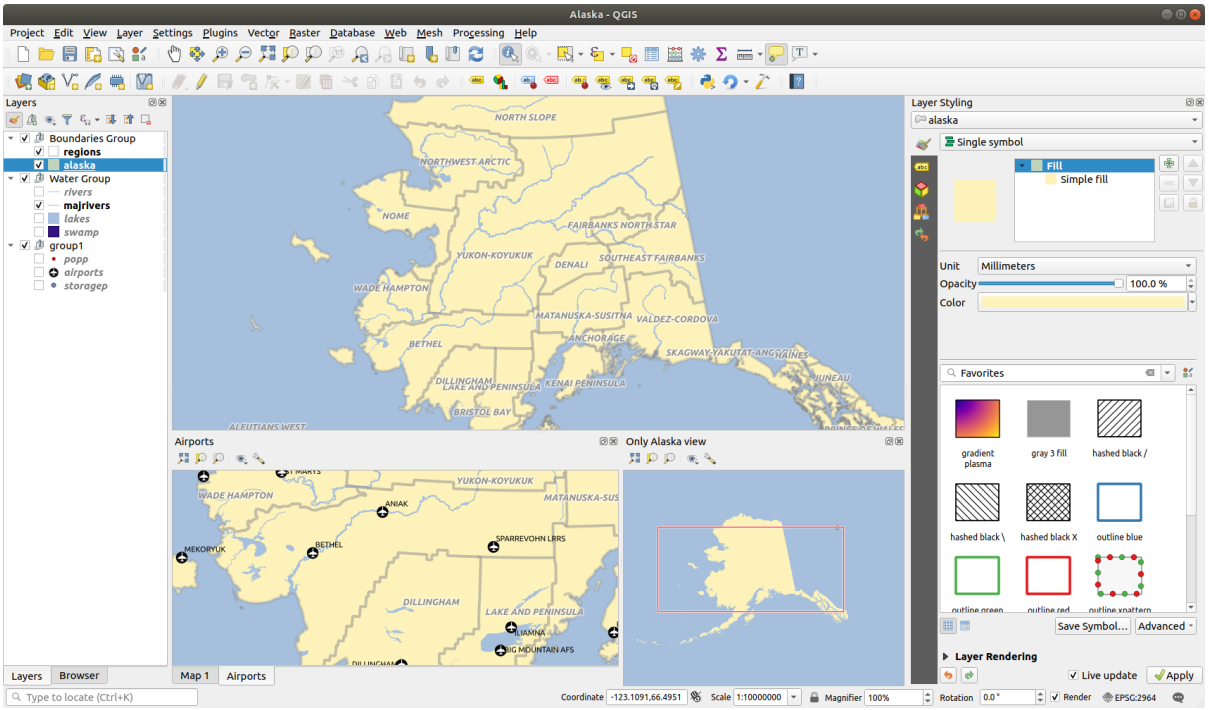


図 11.19: 設定が異なる複数のマップビュー

追加したマップキャンバスの上部には、以下の機能を持つツールバーがあります。

- ビュー内を移動するための 全域表示 、 選択部分にズーム 、 レイヤの領域にズーム
- マップビューに表示する **マップテーマ** を選択するための ビューテーマの設定。(none) に設定されている場合は、ビューは **レイヤ** パネルの変更に従います。
- マップビューの設定のための 表示設定
  - **地図をビューの中心と同期させる** ：縮尺を変えずにマップビューの中心を同期させます。これにより、メインキャンバスの中心に合わせた索引図スタイルや拡大図を作成できます。
  - **ビューを選択と同期させる**：「選択部分にズーム」と同じようにズームします。
  - **縮尺**
  - **回転**
  - **拡大**
  - **縮尺の同期** ：これによってメインマップの縮尺に対する **縮尺係数** を適用でき、例えば縮尺を常に 2 倍にしたビューを作成できます。
  - **注釈を表示する**
  - **カーソル位置を表示する**
  - **メインキャンバスの領域を表示する**
  - **ラベルを表示**：チェックを外すと、表示されているレイヤのプロパティで設定されているかどうかに関係なくラベルを非表示にすることができます。

- *CRS* を変更する...
- ビューの名前を変更...

11.1.9 マップビューのエクスポート

作成したマップは、**印刷レイアウトやレポート** の高度な機能を使用して、さまざまな形式にレイアウトしエクスポートすることができます。また、レイアウトなしで現在の表示を直接エクスポートすることもできます。このマップビューの「スクリーンショット」には、便利な機能がいくつかあります。

現在の表示でマップキャンバスをエクスポートするには、

1. **プロジェクト** → **インポートとエクスポート** を選び、
2. 出力形式に応じて次のいずれかを選択します。
   - **地図を画像にエクスポート**...
   - **地図を** *PDF* **にエクスポート**...

この 2 つのツールには共通のオプションがあります。次のようなダイアログが開きます。

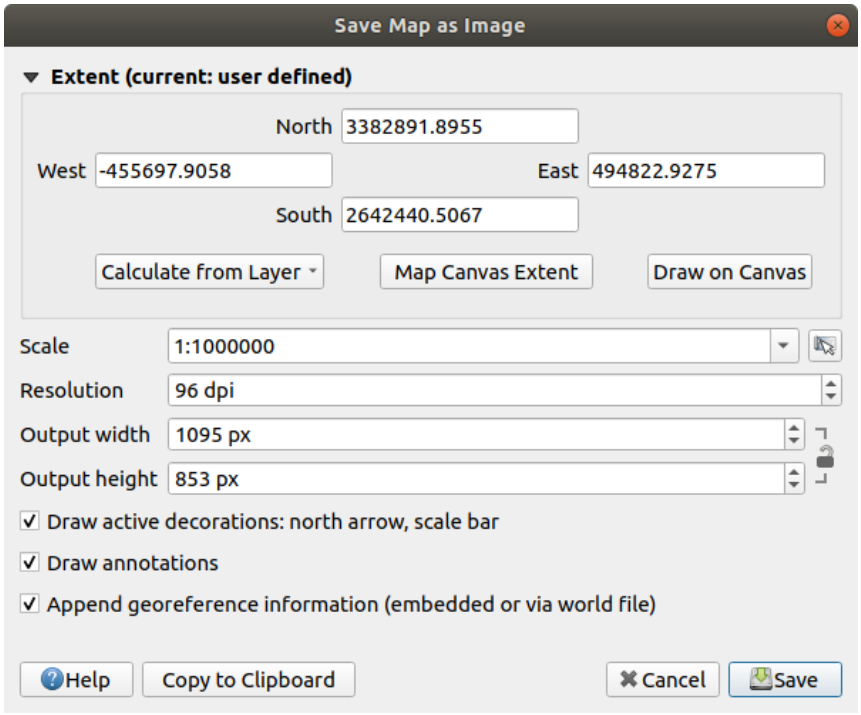


図 11.20: 「地図を画像として保存」ダイアログ

1. エクスポートする **領域** を選択します。現在のビュー（デフォルト）、レイヤの範囲、またはマップキャンバスに描画するカスタムの範囲を選べます。選択した領域の座標がダイアログに表示され、手動で編集できます。
2. 地図の **縮尺** を入力するか、**定義済み縮尺** の中から選択します。 縮尺を変更すると、エクスポートする領域が（中心を基準に）変更されます。
3. 出力の **解像度** を設定します。

4. 画像の **出力の幅** と **出力の高さ** をピクセル単位で設定します。 デフォルトは現在の解像度と領域に基づきますが、変更することもでき、地図の領域が（中心を基準に）変更されます。縦横比は固定することができ、これは出力領域をマップキャンバスに描画するときに特に便利です。
5. ☑ **アクティブな装飾を描く**: 使用中の **地図整飾** （スケールバー、タイトルラベル、グリッド、方位記号など）が地図と共にエクスポートされます。
6. ☑ **注記を描画** ：任意の **注記** をエクスポートします。
7. ☑ **地理参照情報を追加**: 出力形式に応じて、同じ名前のワールドファイル（出力が PNG 画像ならば PNGW 、 JPG 画像ならば JPGW 等の拡張子を持つファイル）が画像と同じフォルダに保存されます。PDF 形式では、PDF ファイルの中に情報が埋め込まれます。
8. PDF にエクスポートする場合は、 **地図を *PDF* にエクスポート**... ダイアログ内で使用できる追加オプションがあります。

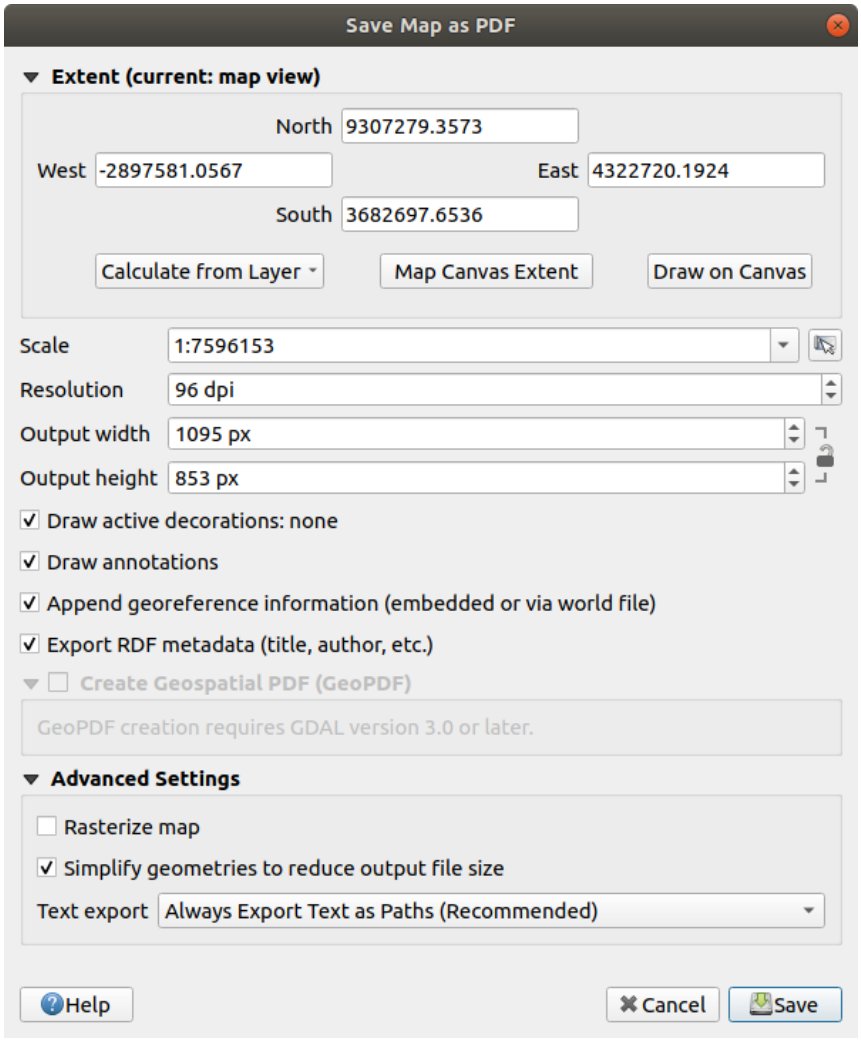


図 11.21: 「PDF 形式で地図を保存」ダイアログ

- ☑ ***RDF* メタデータのエクスポート（*title, author* など）**：タイトル、著者、日付、説明などをエクスポートします。
- ☐ **ジオ *PDF*（*GeoPDF*）を作成**: ジオリファレンスされた PDF ファイル を作成します。以下の設定ができます。

  - **形式** : GeoPDF フォーマットの選択
  - **ベクタ地物情報を含める**: 地図に表示される地物のすべてのジオメトリと属性情報を出力 GeoPDF ファイルに含めます。

注釈: GeoPDF ファイルもデータソースとして使用できます。QGIS での GeoPDF サポートの詳細については、 https://north-road.com/2019/09/03/qgis-3-10-loves-geopdf/ を参照してください。

- **地図をラスタ化する**
- **ジオメトリを簡略化してファイルを縮小する** ：地図をエクスポートする際に、エクスポート先の解像度では区別することができない頂点を削除することで、ジオメトリが簡略化されます（たとえば、エクスポート解像度が `300 dpi` ならば、 `1/600` インチ よりも近い頂点は削除されます)。これにより、出力ファイルのサイズと複雑さを減少させることができます（非常に大きなファイルは他のアプリケーションで読み込みに失敗する可能性があります)。
- **テキスト出力** の設定：これは、テキストラベルを適切なテキストオブジェクトとして出力（**テキストを常にテキストオブジェクトとして出力**）するか、パスのみとして出力（**テキストを常にパスとして出力**）するかを制御します。テキストオブジェクトとして出力する場合、外部アプリケーション（Inkscape など）で通常のテキストとして編集が可能です。ただし、副作用としてレンダリング品質が低下し、さらにテキストにバッファ等の特定の設定がなされていると、レンダリングに問題が発生します。このため、テキストをパスとして出力することを推奨します。

9. **保存** をクリックして、ファイルの場所、名前、形式を選択します。

   地図を画像にエクスポートする場合は、上の設定による出力結果を **クリップボードへコピー** し、LibreOffice や GIMP 等の別のアプリケーションに地図を張り付けることもできます。

11.2 3D マップビュー

3 次元可視化サポートは 3D マップビューによって提供されます。 **ビュー** → ***3D* マップビュー** メニューから、3D マップビューの作成や管理を行ったり、3D マップビューを開くことができます:

1. **新規 *3D* マップビュー** をクリックすると、新しい 3D マップビューを作成できます。ドッキング可能なフローティングの QGIS パネルが現れます（ ***3D* マップビューダイアログ** 参照)。これは 2D のメインマップキャンバスと同じ範囲とビューを持っており、このビューを 3 次元に変化させるためのナビゲーションツール群があります。
2. ***3D* マップビューを管理** をクリックすると、3D マップビューマネージャが表示されます。ここでは、3D マップビューの表示、複製、削除や名前の変更ができます。
3. 1 つ以上の 3D マップビューを作成すると、ビューが ***3D* マップビュー** のメニュー内にリスト表示されます。これをクリックすることで、表示・非表示を切り替えられます。たとえビューが非表示であっても、プロジェクトを保存することで、3D マップビューは保存されます。

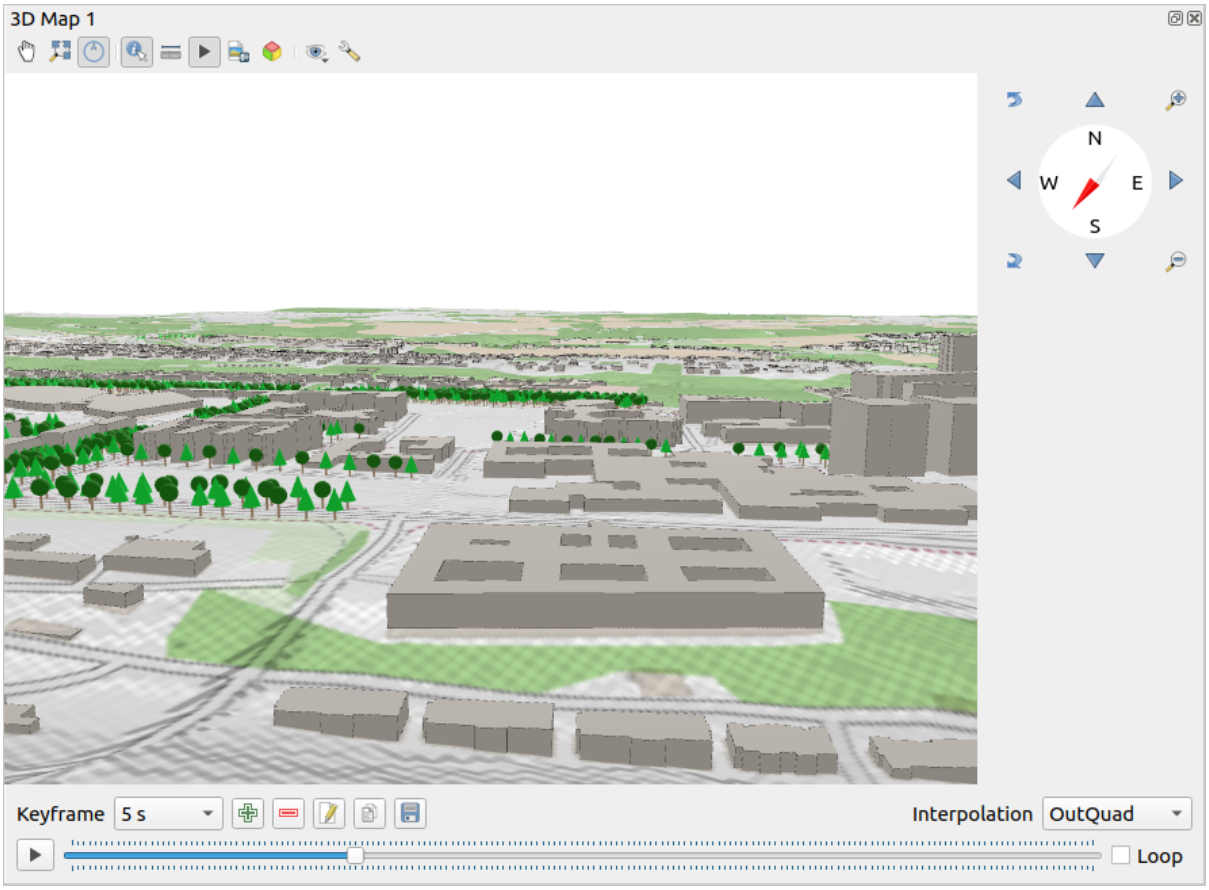


図 11.22: 3D マップビューダイアログ

3D マップビューパネルの上部には、以下のツールがあります。

- カメラコントロール: カメラの角度と方向を維持しながらビューを移動させます
- 全域表示: ビューをレイヤ全体の範囲にサイズ変更します
- 画面操作を切り替え: （マップビューの操作を簡単にするための）ナビゲーションウィジェットを表示/非表示にします
- 地物情報表示: クリックされた地点の地形情報、もしくはクリックされた 3 次元地物の情報を表示します（詳細については **地物の識別** を参照）
- 計測ライン: ポイント間の水平距離を測定します
- アニメーション: **アニメーションプレーヤー** ウィジェットを表示/非表示 にします
- 画像として保存...: 現在のビューを画像ファイル形式でエクスポートします
- 3D シーンの出力 : 現在のビューを 3D シーン（ `.obj` ファイル）として出力し、Blender 等のアプリケーションで後処理できるようにします。地形とベクタ地物が 3D オブジェクトとしてエクスポートされます。エクスポート設定には以下のものが含まれ、レイヤの **プロパティ** やマップビューの **設定** を上書きします。
  - **シーン名** と出力先 **フォルダ**

  - **地形解像度**
  - **地形テキスチャの解像度**
  - **モデルスケール**
  - **エッジのスムージング**
  - **法線をエクスポート**
  - **テキスチャをエクスポート**

- ビューテーマの設定：定義済みの **マップテーマ** から、から、マップビューに表示するレイヤの組み合わせを選択できます。
- オプション メニューには、以下のショートカットがあります:
  - **影** の表示や、 **アイドーム照明（*EDL*）** 、 **環境光遮蔽（*ambient occlusion*）** といった 3D レンダリングへの視覚効果の追加
  - ビューの同期（ *2D* **地図ビューが** *3D* **カメラに従う** や、 *3D* **カメラが** *2D* **地図ビューに従う** ）
  - *2D* **地図ビューにカメラの視界を表示**
  - 設定 ：3D マップビューの **設定**
- 3D マップビューを合体: ドッキング可能なウィジェットからトップレベルウィンドウへと切り替えます

11.2.1 シーン設定

3D マップビューはデフォルトの設定で開かれますが、設定のいくつかはカスタマイズが可能です。設定を変更するためには、3D キャンバスパネルの上部にある オプション メニューを展開し、 **設定** ボタンを押すことで、 *3D* **コンフィグレーション** ウィンドウを開きます。

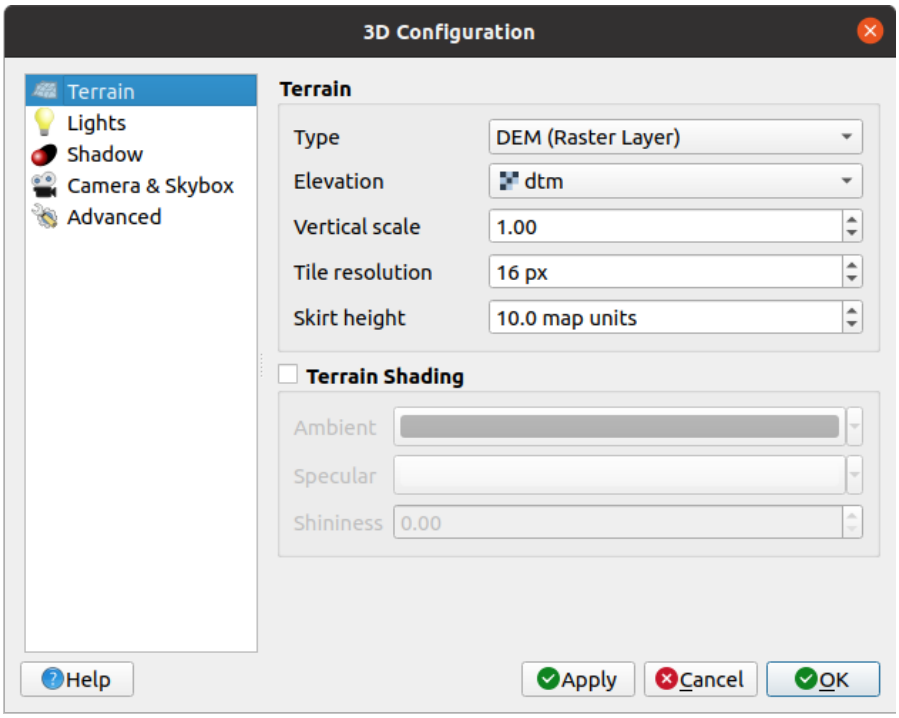


図 11.23: 3D コンフィグレーションダイアログ

3D コンフィグレーションウィンドウには、3D シーンを微調整するためのさまざまなオプションがあります。

地形

- **地形** ：まず、3D ビューの地形は地形タイルの階層によって表現されることに注意してください。カメラが地形に近づくと、十分なディテールを持たない既存のタイルはよりディテールの細かい小さいタイルに置き換えられます。各タイルは、標高ラスタレイヤに由来するメッシュジオメトリと、2D マップレイヤに由来するテクスチャを持っています。
  - 地形は以下の **型** があります。
    - **平らな地形**
    - 読み込まれた *DEM* **ラスタ**
    - **オンライン** サービスは、Mapzen ツールによって作成された 標高タイル を読み込みます。-- 詳細については https://registry.opendata.aws/terrain-tiles/ を参照してください。
    - 読み込んだ **メッシュ** データセット
  - **高さ** ：地形の生成に使用するラスタレイヤまたはメッシュレイヤを選択します。ラスタレイヤには標高を表すバンドが含まれている必要があります。メッシュレイヤは、頂点の Z 値が使用されます。
  - **鉛直スケール**: 垂直軸のスケール係数です。スケールを大きくすると、地形の高さが誇張されます。

  - **タイル解像度**: 各タイルに使用する地形ラスタレイヤのサンプル数です。値が 16 の場合、各タイルのジオメトリは 16x16 の標高サンプルで構成されます。数値を高くするとより詳細な地形タイルが作成されますが、レンダリングが複雑になります。
  - **スカートの高さ**: 地形のタイル間に小さな亀裂が見られることがあります。この値を上げると地形タイルの周りに垂直方向の壁（"スカート"）が追加され、亀裂を隠すことができます。
  - **オフセット**: 地形を上下に移動します。例えば、シーン内の他のオブジェクトのグラウンドレベルに応じて地形の標高を調節するために使います。

    これは、地形の高さとシーン内のレイヤの高さの間に相違がある場合に便利です（例：相対的な垂直方向の高さのみを持つ点群など）。この場合、地形の高さをシーン内のオブジェクトの高さと一致するように手動で調整すると、ナビゲーションの操作性を向上できます。

- メッシュレイヤを地形として使用する場合には、 **三角形設定** （ワイヤフレームを表示、スムース三角形、詳細水準）と、 **レンダリング色設定** （単一色または **カラーランプ** に基づく色）の設定ができます。詳細は **メッシュレイヤの** *3D* **ビュープロパティ** のセクションを参照してください。
- **地形シェーディング**: 地形のレンダリング方法を選択できます。
  - シェーディングが無効 - 地形の色はマップテクスチャからのみ決定されます
  - シェーディングが有効 - 地形の色は Phong のシェーディングモデルを使用して決定されます。マップテクスチャ、地形の法線ベクトル、シーンの光源、地形マテリアルの **環境光の色** と **鏡面光の色** 、 **シャイネス（輝き）** が考慮されます。

光源

光源 タブでは、 メニューをクリックして以下の光源を追加できます。

- 最大 8 つの **点光源** ： 空間に広がる光の球のように、全ての方向に光を放ちます。光源に近い物体は明るく、遠いものは暗くなります。点光源は設定位置（ *X* 、 *Y* 、 *Z* ）と、 **色** 、 **強度** 、 *Attenuation* **（減衰）** を持ちます。
- 最大 4 つの **方向光源** ：これは、オブジェクトから非常に遠く離れた場所にある巨大なフラッシュライトからの光を再現したもので、（例えば太陽のように）常に集中していて弱まることがありません。この光源はある一つの方向に平行な光線を放出しますが、光は無限遠まで届きます。方向光源は **方位角（** *Azimuth* **）** を指定して回転させることができ、 **高度** 、 **色** 、 **強度（** *Intensity* **）** の設定があります。

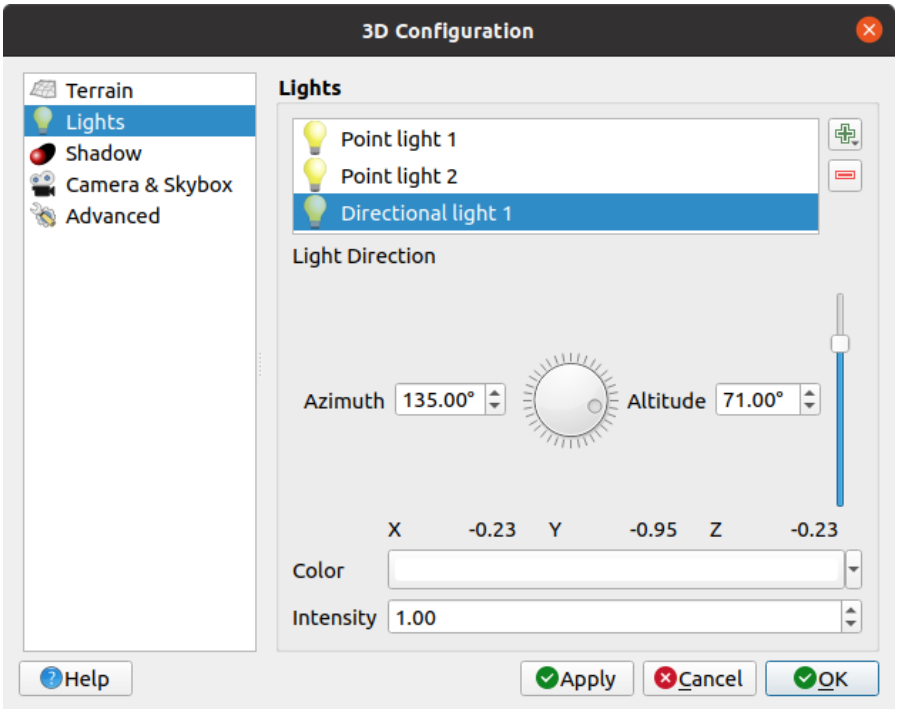


図 11.24: 3D 光源コンフィグレーションダイアログ

影

☐ **影を表示** をチェックすると、シーンに影を表示します。次の設定があります：

- **方向光源**
- **シャドーレンダリングの最大距離** ：特にカメラが水平方向を向いている場合に、遠すぎるオブジェクトの影をレンダリングしないようにするための距離。
- **シャドーバイアス** ：マップサイズの違いによりエリアの一部が他よりも暗くなるセルフシャドウ効果を回避するための値です。小さいほど良いです。
- **シャドー解像度** ：影の見た目をよりシャープにします。解像度パラメータを非常に大きくした場合、パフォーマンスが低下することがあります。

カメラと Skybox

このタブでは、カメラ、3D 軸、ナビゲーションの同期や Skybox といったさまざまなパラメータを制御できます。

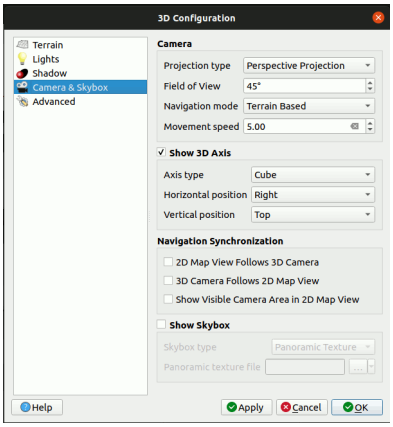


図 11.25: 3D カメラコンフィグレーションダイアログ

- **カメラ** パラメータグループは、 **設定** **オプション** *3D* ダイアログで設定した **デフォルトカメラ設定** の一部を上書きできます。
- *3D* **軸を表示** にチェックを入れると、3D 軸ツールを有効化します。このパラメータグループは、軸のタイプや位置を設定できます。
  - **座標参照系** タイプを使用すると、直交する軸で表現されます。
  - *Cube* タイプを使用すると、3D キューブで表現されます。キューブの面は、カメラのビューの変更に使用できます: 例えば、 **北** の面をクリックすると、カメラが北側から見るように設定されます。

Tip: 3D 軸を右クリックすると、カメラの位置やタイプ、ビューの設定を素早く行えます。

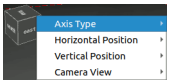


図 11.26: 3D 軸のコンテキストメニュー

- **ナビゲーションを同期** パラメータグループは、2D 地図ビューを 3D カメラ位置に同期させたり、または 3D カメラ位置を 2D 地図ビューに同期させたり、あるいは双方向に同期させたりするオプションを追加します。一番下のオプションは、2D マップビュー上に 3D カメラで見えている範囲を表示させます。
- *Skybox* **を表示** にチェックを入れると、シーンのスカイボックスレンダリングを有効化できます。スカイボックスのタイプには、以下があります：
  - **パノラマテキスチャ** 単一のファイルで、360 ° の視界に適用されます
  - **面（***Distinct faces***）** シーンを囲む箱の 6 つの側面それぞれのテクスチャファイルを指定します

  スカイボックスのテクスチャの画像ファイルには、ディスク上のファイル、リモート URL、プロジェクトに埋め込まれたファイル（ **詳細はこちら** ）が使用できます。

詳細設定

- **地図タイルの解像度**: 地形タイルのテクスチャとして使用される 2D 地図画像の幅と高さ。256px は、各タイルが 256x256 ピクセルの画像にレンダリングされることを意味します。数値が高いほど詳細な地形タイルが作成されますが、レンダリングが複雑になります。
- **最大 画面誤差**: 地形タイルをより詳細なタイル（または簡略なタイル）と交換する際のしきい値を指定します。すなわち、3D ビューがより高品質なタイルにどれだけ早く切り替えるかの値です。数値が低いほど、レンダリングの複雑さを犠牲にしてもシーンの詳細度が高くなります。
- **最大 地上誤差**: タイルをより詳細なタイルへと分割する操作が停止する地形タイル解像度の値です（タイルを分割してもそれ以上のディテールが得られません）。この値はタイルの階層の深さを制限します。値が小さいと階層が深くなり、レンダリングが複雑になります。
- **ズームレベル**: ズームレベルの数を表示します（マップタイルの解像度と最大地上誤差に依存します）。
- ☐ **ラベルを表示**: マップラベルのオン/オフを切り替えます
- ☐ **地図タイル情報を表示**: 地形タイルに境界線とタイル番号を表示します（地形の問題のトラブルシューティングに役立ちます）
- ☐ **バウンディングボックスを表示する**: 地形タイルの 3D バウンディングボックスを表示します（地形の問題のトラブルシューティングに役立ちます）
- ☐ **カメラのビューセンターを表示する**
- ☐ **カメラの回転中心を表示**
- ☐ **ライト源を表示** ：光源の原点に球体を表示し、シーンコンテンツに対する光源の移動や配置を容易にします
- ☐ *FPS* **を表示**
- ☐ **デバッグオーバーレイを表示**: デバッグとプロファイルに関する便利な情報を表示するビジュアルオーバーレイです。これによって特にフレームグラフやシーングラフを素早く確認できます。
- ☐ **アイドーム照明（*EDL*）を表示**: 奥行きの感覚を高めるポストプロセス効果です。各ピクセルの奥行き（カメラからの距離）は隣接ピクセルの奥行きと比較され、奥行きの差に応じて強調表示され、エッジが目立つようになります。これはシーン全体に影響し、**環境光遮蔽（*Ambient Occlusion*）**と組み合わせることができます。以下のパラメータを制御できます:
  - **照明の強度**: コントラストを増加させ、より奥行き間隔を高めます
  - **照明の距離**: 使用するピクセルの中心ピクセルからの距離を表します。エッジを太くする効果があります。
- スクリーン空間への ☐ **環境光遮蔽（*Ambient Occlusion*）** の追加: 環境光に当たりにくい部分により暗い影を適用するポストプロセス効果で、こちらも奥行き感を高めます。これはシーン全体に影響し、**アイドーム照明** と組み合わせることができます。以下のパラメータを制御できます:
  - **半径**: 環境光遮蔽を計算する距離
  - **強度（*Intensity*）**: 効果の強さ（値が大きいほど暗くなる）
  - **遮蔽閾値（*Occlusion Threshold*）**: 隣接点がいくつ遮蔽されると効果が現れるか（50%より低い値の場合、アウトプットは暗くなるが、遮蔽の範囲は広くなることがある）

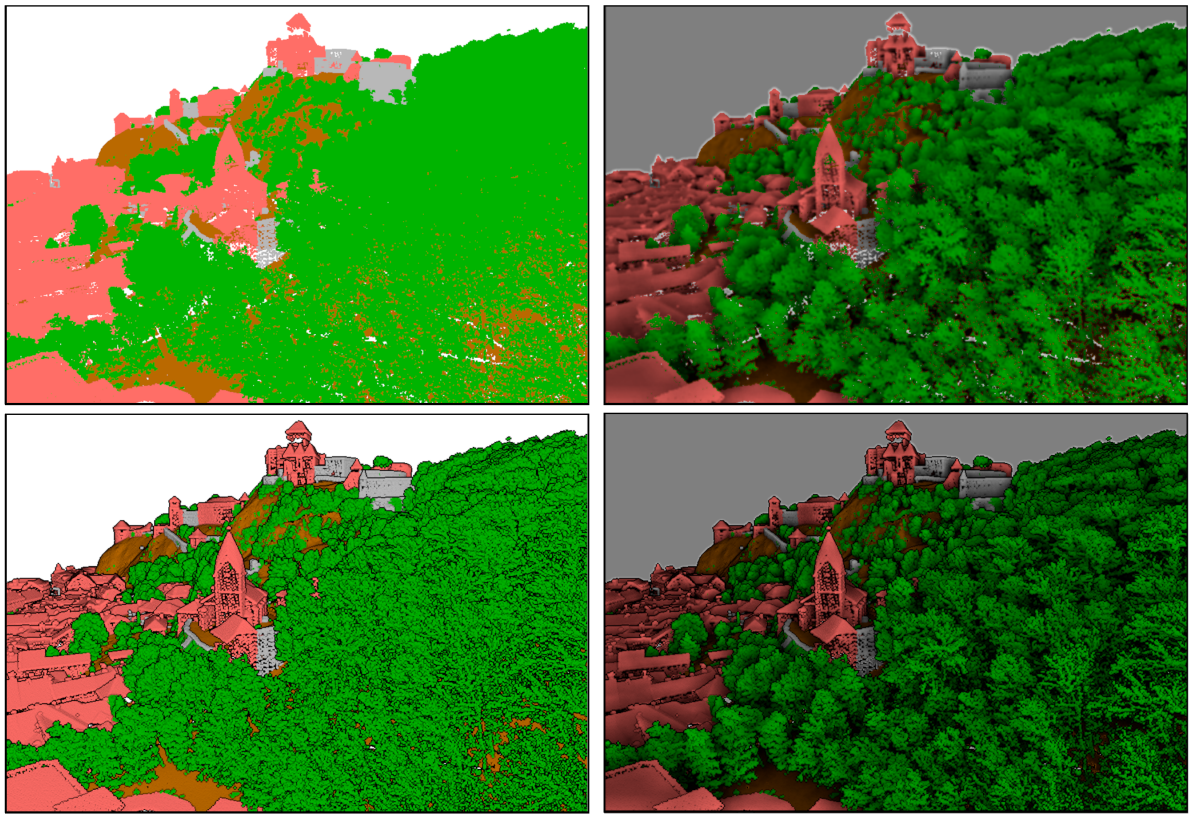


図 11.27: アイドーム照明（EDL）および環境光遮蔽（SSAO）を使用した 3D マップの点群レンダリング
左上から順に、両方なし -- SSAO のみ -- EDL のみ -- SSAO と EDL 両方

- **シャドーマップのデバッグ**: 影に使用する光源の視点から、シーンを赤と黒のイメージとしてレンダリングします（トラブルシューティング用）。ウィジェットは 3D マップビューに比例した **大きさ** に設定され、指定した **コーナー** にドッキングされます。
- **深度マップのデバッグ**: シーンの深度マップをカメラに近いピクセルは暗い色の画像としてレンダリングします（トラブルシューティング用）。ウィジェットは 3D マップビューに比例した **大きさ** に設定され、指定した **コーナー** にドッキングされます。

11.2.2 ナビゲーションオプション

マップビューを 3D で見るには、

- 地形を傾けます（ウィンドウの中心を通る水平軸を中心に回転させます）
  - Tilt up （上に傾ける） と Tilt down （下に傾ける） ツールボタンを押す
  - `Shift` キーを押しながら、上/下キーを押す
  - マウスの中央ボタンを押したまま、マウスを前後にドラッグする
  - キーボードの `Shift` キーとマウスの左ボタンを押しながら、マウスを前後にドラッグする
- 地形を回転させます（ウィンドウの中心を通る垂直軸を中心に回転させます）
  - ナビゲーションウィジェットのコンパスを見たい方向に向ける

  - Shift キーを押しながら、左/右キーを押す
  - マウスの中央ボタンを押したまま、マウスを左右にドラッグする
  - キーボードの Shift キーとマウスの左ボタンを押しながら、マウスを左右にドラッグする
- カメラ位置（とビューの中心）を変更し、水平方向に移動します
  -  カメラコントロール ボタンを有効にした状態で、マウスの左ボタンを押しながらドラッグする
  - ナビゲーションウィジェットの方向矢印ボタンを押す
  - キーボードの上/下/左/右キーを押して、カメラをそれぞれ前/後/左/右に移動させる
- カメラの高度を変更します：キーボードの Page Up / Page Down キーを押す
- カメラの向きを変更します（カメラはその位置を維持したまま、ビューの中心が移動します）
  - Ctrl キーを押しながら矢印キーを押して、カメラを上下左右に回転させる
  - キーボードの Ctrl キーとマウスの左ボタンを押しながら、マウスをドラッグする
- ズームインとズームアウト
  - ナビゲーションウィジェットの対応する  拡大 ・  縮小 ツールボタンを押す
  - マウスホイールをスクロールする（Ctrl キーを押した状態だとより細かいズーム）
  - マウスの右ボタンを押しながらドラッグして、拡大（下にドラッグ）・縮小（上にドラッグ）する

カメラビューをリセットするには、3D キャンバスパネルの上部にある  全域表示 ボタンをクリックします。

11.2.3 アニメーションの作成

アニメーションはキーフレーム（特定の時刻のカメラ位置）の集合に基づいています。アニメーションを作成するには、

1.  アニメーション ツールをオンに切り替え、アニメーションプレイヤーウィジェットを表示させます
2.  キーフレームを追加 ボタンをクリックして、**キーフレーム時間** を秒単位で入力します。**キーフレーム** コンボボックスには、キーフレーム時間の集合が表示されます。
3. ナビゲーションツールを使用してカメラを移動させ、カメラ位置を現在のキーフレーム時間に関連付けます。
4. 上の手順を繰り返して、必要な数のキーフレーム（時間とカメラ位置）を追加します。
5.  ボタンをクリックして、アニメーションをプレビューします。QGIS は設定したキーフレーム時間のカメラ位置/回転 を使用してシーンを生成し、キーフレーム間のカメラ位置/回転を補間します。アニメーションにはさまざまな **内挿** モードが利用可能です（例えば Linear（線形）、 InQuad、OutQuad、InCirc など -- 詳細は https://doc.qt.io/qt-5/qeasingcurve.html#EasingFunction-typedef を参照してください）。

アニメーションは、時間スライダーを動かすことでもプレビューできます。 **ループ** のチェックボックスにチェックを入れると、 ▶ ボタンをクリックしてアニメーションを停止するまでは、アニメーションを繰り返し再生します。

アニメーション出力フレーム を使用して、シーンを表す一連の画像を生成します。出力ファイル名の **テンプレート** や **出力フォルダ** のほか、**フレーム数/秒**、 **出力の幅**、 **出力の高さ** を設定することができます。

11.2.4 3D ベクタレイヤ

標高値を持つベクタレイヤは、ベクタレイヤのプロパティの *3D* **ビュー** セクション内の *Enable 3D Renderer* をチェックすることで、3D マップビューに表示することができます。 3D ベクタレイヤのレンダリングを制御するためのオプションが多数用意されています。

11.3 標高断面図ビュー

標高断面図 パネルは側面図を作成するためのツールで、線に沿った標高を可視化することができます。このツールはベクタ、ラスタ、メッシュ、点群レイヤをサポートします。データは 2 次元または 3 次元の形式です。

標高断面図ビューを追加するには、 **ビュー** **標高断面図** メニューにアクセスします。必要に応じていくつでも標高断面図ビューを追加でき、ビューはドッキング可能で、互いに積み重ねることも、フローティングにすることもできます。

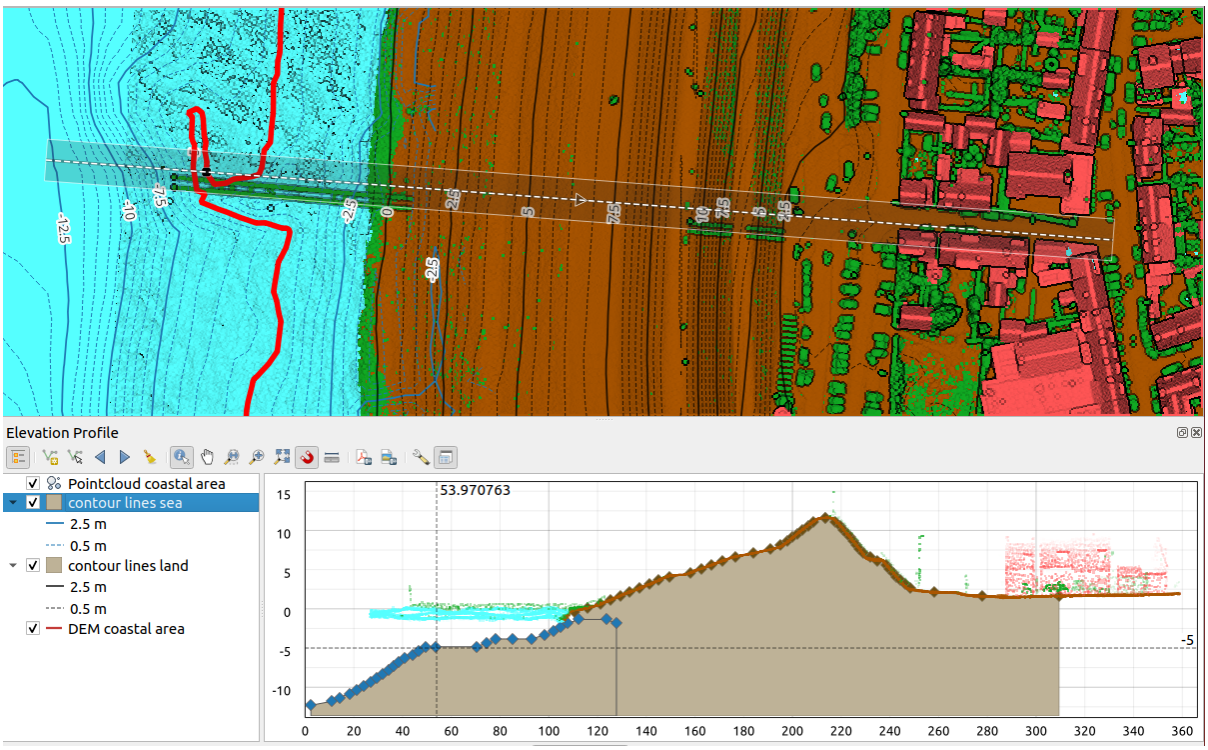


図 11.28: メインマップキャンバスの下部に埋め込まれた標高断面図ダイアログ
図は海岸線と海岸堤防の背後にある町を示しています。標高断面図の線に適用する許容誤差を大きくすると、より多くの点群地物が返されます。

11.3.1 インターフェース

標高断面図 パネルの上部には、以下のツールを有するツールバーがあります:

表 11.4: 標高断面図ビューのツールバー

ツール	ショートカット	説明
 レイヤツリーを表示		標高断面図ビューにおけるレンダリングを設定するためのプロジェクトレイヤのリストの表示・非表示を切り替えます。
 曲線キャプチャ		プロファイル曲線を表す線をマップキャンバス上でインタラクティブに描画します。
 地物から曲線をキャプチャ		マップキャンバス上の既存のライン地物を選択し、そのラインに沿ってプロファイル曲線を生成します。
 やや左	Ctrl+Alt	地図上のキャプチャ曲線を少しずつ左に動かします（例：最適な標高プロファイル曲線を見つけるため）。
 やや右	Ctrl+Alt	地図上のキャプチャ曲線を少しずつ右に動かします（例：最適な標高プロファイル曲線を見つけるため）。
 クリア		プロファイル曲線と **標高断面図** ビューに表示されたプロットをすべて削除します。
 地物情報を表示		シングルクリックで、または長方形をクリック＆ドラッグで、プロットキャンバス内の地物情報を表示します。結果は標準の **地物情報結果** ドックに表示されます。
 パン	Space	クリック＆ドラッグでプロットキャンバスを移動させます。中央マウスボタンでも操作できます。
 X 軸ズーム		縦方向の縮尺はそのまま、横軸に沿って拡大します。
 ズーム	Ctrl+Spa	プロット上をクリックで、または長方形をクリック＆ドラッグで、プロットを拡大します。 `Alt` キーを押しながらクリックすると、反対に縮小します。
 全域表示		**標高断面図** ビューをキャプチャ曲線の範囲に合わせて拡大縮小します。
 スナップを有効化		標高断面図ビュー内にプロットされた地物の辺や頂点にスナップします。座標の正確な取得や距離の測定に便利です。
 距離を計測		水平距離と垂直距離を計測します。
 PDF として出力		プロットを PDF へ（高品質なベクタオブジェクトとして）エクスポートします。
 画像として出力		プロットをいくつかの画像形式でエクスポートします。
 オプション		標高断面図曲線の設定にアクセスします。
 標高断面ビューを合体		ビューのドック状態とフロート状態を切り替えます。

 **レイヤツリーを表示** ボタンを押すと、左下に **レイヤ** パネルのコピーが表示されます。ただし、これ

は独立したウィジェットで、表示レイヤの組み合わせや重ね合わせ順序は独自に持っています。このウィジェットで、レイヤのレンダリングやプロットキャンバス内での動作を制御できます:

- レイヤ名の左にあるボックスをクリックすると、プロットキャンバスにレンダリングするかどうかを設定できます
- レイヤを上下にドラッグ＆ドロップして、レイヤの順序を変更できます
- 標高断面図ビューでのレイヤのレンダリングスタイルを設定できます：レイヤをダブルクリックするか、右クリックして **プロパティ**... を選択すると、設定のためにレイヤの **標高** プロパティタブが開きます。レイヤ上にカーソルを乗せると、標高の設定に関する概要がツールチップとして表示されます。

レイヤツリーの右にあるプロットキャンバスは、有効になっているレイヤの標高断面図をプレビューできるメインの場所です。キャンバスのベースはメモリ付きのグリッドで、横軸はプロファイル曲線の長さ、縦軸は測定した地物の Z 標高を表します。また、上にあるツールを使用してズーム、パン、距離計測や地物情報表示などの操作ができます。

11.3.2 標高断面図の作成

標高断面図ビューを作成するには、以下のように操作します:

1. **ビュー** **標高断面図** メニューを選択します。 **標高断面図** パネルが開きます。
2. プロファイル曲線を作成します。この線に沿って地形や地物がレンダリングされます。描画ツールを以下から選択します:
   - 曲線キャプチャ: メインマップキャンバスを左クリックして頂点を追加し、右クリックで描画を終了することで作成した線をプロファイル線として使用します
   - または、 地物から曲線をキャプチャ: マップキャンバス上のライン地物をクリックし、プロファイル線として選択します。クリックした点に複数の地物がある場合には、ポップアップメニューが現れ、その中から地物を選択できます。

   ラインのデジタイズに関するあらゆる機能、例えば **スナップオプション** や、 **トレース** 、 **デジタイズのテクニック** 、 **高度なデジタイズパネル** などを利用することができます。

   ここで、プロットキャンバスはいくつかの地物をレンダリングするかもしれません。
3. 次のステップは、可視化したいレイヤの標高プロパティの設定です。
   1. レイヤツリーを表示 ボタンを押して、レイヤのリストを表示します。
   2. 興味のあるレイヤの表示/非表示を切り替えます。これらのレイヤは標高断面図ビューにレンダリングされるだけで、選択するレイヤはメインの **レイヤ** パネルとは異なったものにできます。
   3. レイヤ名をダブルクリックするか、右クリックして **プロパティ** を選択します。レイヤの **標高** プロパティタブが開きます。ここで各地物または地形が標高断面図ビューでどのようにレンダリングされるかを設定します。使用できるプロパティはレイヤの種類によって異なります：
      - **ラスタ標高プロパティ**
      - **ベクタ標高プロパティ**

- **点群標高プロパティ**
- **メッシュ標高プロパティ**

標高プロパティが設定されると、標高断面図ビューは設定したプロファイル曲線と交差するアクティブなレイヤの地形または地物のレンダリングを開始します。

4. オプション ドロップダウンメニューからは、**許容範囲** の値の設定ができます。この値は、メインマップキャンバスに表示される標高プロファイル線の周りに平坦なバッファを作成するために使用されます。このバッファに重なる可視状態のポイント地物は、プロットキャンバスにキャプチャされます。

11.3.3 標高断面図ビューの操作

標高断面図のラインを作成すると、プロットキャンバスはその全範囲にズームします。X 軸はプロファイルの長さ、Y 軸はキャプチャした標高の最大値・最小値の範囲となります。どちらの軸も地図単位です。

標高断面図ビュー内でマウスポインターを動かすと、2 つの交差する点線が表示されます:

- 横線は高さの情報
- 縦線は標高断面図ラインの始点からの距離

標高断面図ビュー内でマウスポインターを動かすと、メインマップキャンバスで標高断面図のラインに沿って移動する黒い点があることも確認できます。ラインの中間には、方向を表示する矢印があります。

メインマップキャンバスと同様に、QGIS はプロットキャンバス上をナビゲートする手段を提供します:

- パン は、標高断面図の範囲を好きな方向に移動させるのに使います。 Space キーを押しながらマウスを移動させることでも、プロットキャンバスの範囲を平行移動できます。
- X 軸ズーム は、縦軸（標高）の縮尺を保ったまま、横軸に沿って拡大するのに使います。左クリックすると、クリックした点を X 軸の中心として軸に沿って引き伸ばします。また、長方形をドラッグすると、X 軸に沿って長方形の幅にプロットを引き伸ばします。 Alt キーを押しながら X 軸ズーム を使用すると、X 軸方向に縮小します。
- ズーム は、（左クリックで）ある一点に拡大する、または（範囲を長方形ドラッグで）ある範囲に拡大するのにつかいます。 Alt キーを押しながら ズーム を使用すると、反対に縮小します。 Ctrl キーと組み合わせると、より滑らかに拡大・縮小することができます。
- 全域表示 は最初に使用したデフォルトのズームレベルで、表示する地物すべてに関する標高断面図のラインの全範囲を表示します。ズームレベルのリセットに使用します。

プロットキャンバスに表示された要素とのインタラクションもあります:

- スナップを有効化 ボタンを押すと、ポイントや地物の頂点やエッジを正確にとらえることで、正確な距離計測や座標取得ができます。
- 地物情報を表示 は、レイヤツリーで表示されているレイヤの地物の識別に使用します。標高断面図ビュー内でいくつかの地物にまたがって長方形をドラッグすると、それらの地物をクエリできます。対応した形式（例：ベクタレイヤや点群など）ならば、その地物がメインマップキャンバスで強調表示されます。

- 距離を計測: プロットキャンバス内で 2 点をクリックまたは選択すると、2 点間の水平 **距離** 、 **標高** 差、および **合計長** を地図単位で表示します。
- やや左 と やや右 は、マップキャンバス内の標高断面図のラインの位置を左右に移動させるのに使います。プロットキャンバスは再描画され、標高断面図のラインバッファと重なる地物や地形が表示されます。ラインの左右への移動量には、 オプション メニューの **許容範囲** の値を使用します。

警告: 現時点では、標高断面図ビューやプロジェクトを閉じると、ビューがプロジェクトから削除されてしまいます。

更なる詳細については、Nyall Dawson 氏によるプレゼンテーション QGIS elevation profile/cross section tool -- a deep dive! をご覧ください。

第12章　一般ツール

12.1 コンテキストヘルプ

特定のトピックでヘルプが必要な場合は、多くのダイアログで利用できるようになっている **ヘルプ** ボタンを押すと、このユーザマニュアルの対応するページにアクセスできます。ただし、サードパーティのプラグインは専用の Web ページにアクセスする場合があることに注意してください。

12.2 パネル

QGIS はデフォルトで多くのパネルを利用することができます。いくつかのパネルは以下に解説がありますが、その他はユーザマニュアルの別の場所に説明があるものもあります。QGIS に用意されているデフォルトのパネルの完全なリストは **ビュー**　**パネル**　で参照でき、 **パネル** に説明されています。

12.2.1 レイヤパネル

レイヤ パネル（地図凡例 とも呼ばれます）は、プロジェクト内のすべてのレイヤを一覧表示し、その可視性を管理したり、マップの形を整えるのに役立ちます。このパネルは `Ctrl+1` を押すことで表示・非表示を切り替えることができます。

レイヤ パネルの上部にあるツールバーによって、以下のことができます：

-  レイヤのスタイルパネルを開く（F7）: **レイヤスタイルパネル** の表示、非表示を切り替えます。
-  グループ追加: **グループやレイヤの操作** を参照してください
-  地図テーマを管理: レイヤの可視設定を管理し、様々な **地図テーマ** にレイヤをアレンジメントします。
-  凡例ツリー内のレイヤをフィルタリングします：
  - **コンテンツで凡例を絞り込む**: レイヤが可視状態に設定されていて、かつ地物が現在の地図キャンバス内に表示されているレイヤのみが、レイヤパネルでスタイルが表示されるようになります。それ以外の場合は、一般的な NULL シンボルがレイヤに適用されます。このオプションは、関心のある領域に存在するのがどのレイヤのどの地物なのかをレイヤシンボルに基づいて識別するのに便利な方法です。
  - **プライベートレイヤを表示**: プロジェクトの設定を変更せずに **プライベートレイヤ** を **レイヤ** パネル内に表示したり、操作するのに便利なショートカットです。

- 条件式による凡例フィルタ：式を適用して、条件を満たす地物がないスタイルを選択したレイヤツリーから削除します。これは、別のレイヤの特定の領域/地物内にある地物を強調表示するために使用できます。ドロップダウンリストから、現在適用されている式を編集およびクリアできます。
- すべて展開する と すべて折りたたむ は、レイヤパネル内のすべてのレイヤとグループを展開する、あるいは折りたたみます。
- レイヤ/グループを削除 現在選択されているレイヤやグループを削除します。

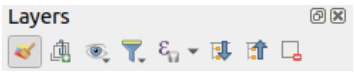


図 12.1: レイヤパネル内のレイヤツールバー

注釈: レイヤパネルを管理するツールは、印刷レイアウトのマップアイテムと凡例アイテムにも使用できます。

地図テーマの設定

地図テーマを管理 ドロップダウンボタンを使用すると、 **レイヤ** パネルの可視性を操作するための便利なショートカットにアクセスできます。

- **すべてのレイヤを表示**
- **すべてのレイヤを隠す**
- **選択レイヤを表示**
- **選択レイヤを隠す**
- **選択レイヤを切り替え**: は、パネル上で最初に選択されたレイヤの表示状態を変更し、その状態を他に選択されたレイヤにも適用します。 また、スペースキー のショートカットからも利用できます。
- **選択レイヤを個別切り替え**: は、選択された各レイヤについて表示状態を変更します
- **選択されてないレイヤを隠す**

地図テーマを管理 メニューでは、レイヤの可視性の単純な制御だけでなく、凡例で **地図テーマ** を構成し、ある地図テーマから別の地図テーマに切り替えたりすることができます。地図のテーマは、現在の地図凡例の **スナップショット** であり、以下のものを記録しています：

- **レイヤ** パネル内で可視状態に設定されたレイヤ
- **さらに** 、各可視レイヤに関する以下の情報：
  - レイヤに適用された **スタイル** への参照情報
  - スタイルのクラスの可視状態、すなわち、 **レイヤパネル** 内でチェックされたノード項目のレイヤ。これは単一定義のシンボルレンダリング以外の **シンボロジ** について適用されます。

- 中に配置されいるレイヤのノードやグループの折りたたみ/展開状態

地図テーマを作成するには、

1. 表示したいレイヤをチェックします
2. レイヤのプロパティ（シンボロジ、ダイアグラム、ラベルなど...）をいつもどおりに設定します
3. 下の方にある **スタイル** メニューを展開し、 **追加**... を押して、 **プロジェクトに埋め込まれた新しいスタイル** として保存します

   注釈: 地図のテーマはプロパティの現在の内容を保存しません。スタイル名への参照のみが保存されるため、このスタイルが有効なときにレイヤに変更を適用する（例えばシンボルレンダリングを変更する）と、テーマは新しい情報で更新されます。

   ---

4. 他のレイヤについても必要なだけ上の手順を繰り返します
5. 該当する場合は、 **レイヤ** パネルのグループや可視レイヤのノードを展開または折りたたみます。
6. パネル上部の 地図テーマの管理 ボタンをクリックし、 **テーマの追加**... をクリックします。
7. 地図テーマの名前を入力し、 *OK* ボタンをクリックします

新しい地図テーマが ドロップダウンメニューの下部にリストされます。

地図テーマは必要な数だけ作成できます。地図凡例の現在の組み合わせ（可視レイヤ、有効なスタイル、地図凡例ノード）が上記で定義された既存の地図テーマ内容と一致しない場合は、 **テーマの追加**... をクリックして新しい地図テーマを作成するか、 **テーマの置き換え** を使用して地図テーマを更新します。アクティブな地図テーマの名前を **現在のテーマの名前変更**... で変更することもできますし、 **現在のテーマを削除** ボタンで地図テーマを削除することもできます。

地図テーマは、あらかじめ設定されたさまざまな地図凡例の組み合わせをすばやく切り替えるのに便利です。リストで地図テーマを選択すると組み合わせが復元されます。設定されたすべてのテーマは印刷レイアウトにおいてもアクセスでき、特定のテーマに基づいて、現在のメインキャンバスのレンダリングとは関係なく、異なる地図アイテムを作成できます（ **地図アイテムレイヤ** を参照）。

レイヤパネルのコンテキストメニューの概要

ツールバーの下部にあるレイヤパネルの主な構成要素は、プロジェクトに追加されたすべてのレイヤを一覧表示するフレームで、オプションでグループ別に整理することもできます。レイヤの隣にチェックボックスがある場合、 **縮尺に応じた表示** が設定されていない限り、その内容はマップキャンバスの範囲に重なって表示されます。レイヤを選択し、凡例内で上下にドラッグすると、Z 順序を変更することができます。Z 順序とは、凡例の上部に近いレイヤほど、凡例の下方に並んでいるレイヤの上に描画されることを意味します。また、レイヤまたはレイヤのグループは、複数の QGIS インスタンスにまたがってドラッグすることができます。

注釈: Z 順序の動作は **レイヤ順序** パネルで上書きすることができます。

パネルで選択した項目に応じて、右クリックで以下のような専用オプションが表示されます。

表 12.1: **レイヤ** パネル項目からのコンテクストメニュー

オプション	グループ	ベクタレイヤ	ラスタレイヤ	メ
レイヤ/グループの領域にズーム	✓	✓	✓	
選択部分にズーム		✓		
全体図に表示		✓	✓	
地物の数を表示		✓		
ラベルを表示		✓		
レイヤ/グループをコピー	✓	✓	✓	
レイヤ/グループの名前を変更	✓	✓	✓	
ネイティブ解像度に戻す			✓	
現在の領域に引き伸ばす			✓	
SQL **レイヤの更新...**		✓		
仮想レイヤを編集...		✓		
グループを追加	✓			
レイヤを複製		✓	✓	
レイヤ/グループを削除...	✓	✓	✓	
グループ外に移動		✓	✓	
一番上に移動	✓	✓	✓	
一番下に移動	✓	✓	✓	
グループと全ての親をチェック		✓	✓	
選択レイヤをグループにする		✓	✓	
属性テーブルを開く		✓		
編集モード切り替え		✓		
現在の編集		✓		
フィルタ...		✓	✓	
データソースを変更...		✓	✓	
データソースを修復...		✓	✓	
選択に対するアクション （編集モード時）		✓		
地物の複製		✓		
地物の複製とデジタイズ		✓		
レイヤの縮尺表示を設定...		✓	✓	
表示される縮尺にズーム		✓	✓	
レイヤの *CRS*		✓	✓	
レイヤの *CRS* **をプロジェクトの** *CRS* **に設定**		✓	✓	
（最近使用した *CRS*）**に設定..**				
レイヤの *CRS* **を設定...**		✓	✓	
グループの *CRS* **を設定...**	✓			

表 12.1 – 前のページからの続き

オプション	グループ	ベクタレイヤ	ラスタレイヤ	メ
グループの *WMS* データを設定...	☑			
相互排他的グループ	☑			
グループと全メンバをチェック（*Ctrl* キーを押しながらクリック）	☑			
グループと全メンバのチェックを外す（*Ctrl* キーを押しながらクリック）	☑			
保存...		☑		
エクスポート	☑	☑	☑	☑
名前を付けて保存...			☑	
新規ファイルに地物を保存...		☑		
新規ファイルに選択地物を保存...		☑		
レイヤ定義ファイルとして保存...	☑	☑	☑	☑
***QGIS* レイヤスタイルファイルとして保存...**		☑	☑	☑
スタイル		☑	☑	☑
スタイルのコピー		☑	☑	☑
スタイルの貼り付け	☑	☑	☑	☑
追加...		☑	☑	☑
現在のスタイル名を変更...		☑	☑	☑
シンボルを編集...		☑		
シンボルをコピーする		☑		
シンボルを貼り付ける		☑		
レイヤノートを追加...		☑	☑	☑
レイヤノートを編集...		☑	☑	☑
レイヤノートを削除		☑	☑	☑
プロパティ...		☑	☑	☑

GRASS ベクタレイヤについては、 編集モード切り替え は使用できません。GRASS ベクタレイヤの編集については、 *GRASS* **ベクタレイヤをデジタイズして編集する** のセクションを参照してください。

グループやレイヤの操作

凡例ウィンドウのレイヤは、グループに整理することができます。これにはさまざまな方法があります:

1. アイコンを押して新しいグループを追加します。グループの名前を入力し Enter を押します。それから、既存のレイヤをクリックしてグループにドラッグします。
2. 複数のレイヤを選択し、 アイコンを押します。選択されたレイヤは、自動的に新しいグループに挿入されます。
3. レイヤをいくつか選択し、凡例ウィンドウで右クリックして **選択レイヤをグループにする** を選択します。選択したレイヤが自動的に新しいグループに配置されます。

レイヤをグループ外に移動するには、ドラッグするか、右クリックして **グループ外に移動** を選択します。レイヤはグループ外に移動され、グループの上に配置されます。グループは他のグループ内に入れ子にもできます。レイヤが入れ子にされたグループに配置されている場合、 **グループ外に移動** は、入れ子になった全てのグループの外にレイヤを移動します。

グループまたはレイヤをレイヤパネルの最上位に移動するには、それらを最上位にドラッグするか、 **一番上に移動** を選択します。グループ内に入れ子になっているレイヤでこのオプションを使用すると、レイヤは現在のグループ中で最上位に移動します。 **一番下に移動** オプションは同じ要領で、レイヤやグループを一番下に移動します。

グループのチェックボックスは、グループ内でチェックされたレイヤ全てを 1 クリックで表示・非表示にできます。 `Ctrl` キーを押しながらグループのチェックボックスをクリックすると、グループ内とそのサブグループ内の全てのレイヤも表示・非表示が切り替わります。

`Ctrl` を押しながらチェックされた/されていないレイヤをクリックすると、そのレイヤとすべての親レイヤのチェックを外し/付けます。

相互排他的グループ のオプションを有効にすると、グループ内で同時に表示されるレイヤが 1 つだけとなるようにすることができます。グループ内のあるレイヤが表示状態となると、他のすべてのレイヤは非表示になります。

`Ctrl` キーを押しながら追加のレイヤをクリックすると、複数のレイヤやグループを同時に選択することができます。選択したレイヤを同時に新しいグループに移動させることができます。

また、複数のアイテムを `Ctrl` キーを押しながら選択してから `Ctrl+D` を押すことで、選択されたすべてのレイヤやグループをレイヤのリストから一度に削除することもできます。

インジケータアイコンによるレイヤとグループの詳細情報

状況によっては、 **レイヤ** パネルのレイヤやグループの形式や、横に表示されるアイコンが変わり、そのレイヤ/グループについての詳しい情報が得られます。これらの要素は次のとおりです:

-  レイヤが編集モードであり、データを変更できることを示します
-  編集中のレイヤに未保存の変更があることを示します
-  レイヤに **フィルタ** が適用されていることを示します。アイコンにカーソルをのせるとフィルタ式が表示され、ダブルクリックするとクエリ式を変更できます
-  このレイヤがプロジェクトに **必須の** レイヤであり、削除できないことを示します
-  **埋め込まれたグループやレイヤ** であることを示し、オリジナルのプロジェクトファイルへのパスを表示します
-  プロジェクトファイルを開いたときにデータソースが利用できなかったレイヤであることを示します（ **壊れたファイルパスの取り扱い** を参照）。アイコンをクリックするか、レイヤのコンテキストメニューから **データソースの変更**... エントリを選択して、データソースのパスを更新してください。
-  レイヤが **一時スクラッチレイヤ** であり、このプロジェクトを閉じるとその内容は破棄されることを気付かせます。データの損失を避け、レイヤを恒久的なものにするには、アイコンをクリックして QGIS がサポートする GDAL ベクタ形式のいずれかにレイヤを保存します。

- レイヤが **オフライン編集モード** に使用されていることを表します
- CRS がない/不明なレイヤであることを示します
- 本質的に精度の低い CRS で保存された座標を持つレイヤであることを表します（**対応する設定** を有効にする必要があります）
- キャンバスアニメーションによって制御される時系列レイヤであることを示します
- 関連付けられた **レイヤノート** があるレイヤであることを表します
- 名前が灰色で表示されるのは、マップキャンバスの現在の縮尺が、そのレイヤを表示する縮尺の範囲（**レンダリング** プロパティで設定されている）の外にある場合です。コンテキストメニューの **表示される縮尺にズーム** オプションを選択すると、そのレイヤの最も近い表示スケール範囲にマップをズームすることができます。

レイヤのレンダリングをグループで制御する

グループは、プロジェクトのツリー内でレイヤを構造化する手段ですが、マップのレンダリング時にフラット化された単独のオブジェクトとして扱われることで、その構成レイヤのレンダリング方法に影響することができます。

このようなレンダリングのオプションは、グループが選択されたときに **レイヤスタイル** パネル内で利用できます。シンボロジ タブで、**グループとしてレイヤを描画** をチェックすると、個々のレイヤの代わりに、子レイヤ全体の外観を制御する一連のオプションが有効になります：

- **不透明度**: 他の子レイヤによって隠されている子レイヤの地物は、そのまま隠されており、不透明度は「グループ全体」にのみ適用されます。

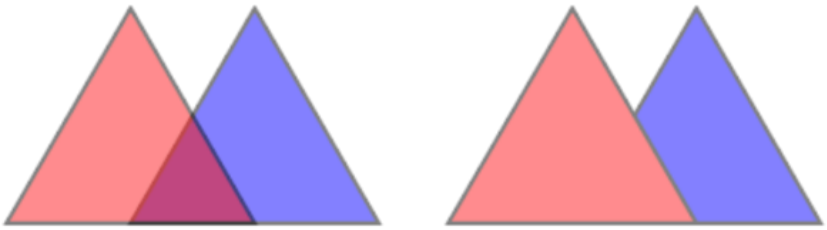


図 12.2: 不透明度をレイヤに設定した場合とグループに設定した場合
左の画像は、2 つのレイヤーを不透明度 50 %でレンダリングしたものです（下にある地物は見えますが、上にある 50 %の赤い地物によって半分マスキングされています）。2 番目の画像は、グループの不透明度を設定した結果です（下にある青の子レイヤーの一部が、上にある赤のレイヤーによって完全に隠されており、その結果が不透明度 50 %でレンダリングされています）。

- **混合モード**: 不透明度と同じように、グループ全体に :ref:`混合モード <blend-modes>`（乗算、オーバーレイ...）を設定すると、まず子レイヤの地物がフラット化され、上のレイヤが下のレイヤを覆い隠します。レンダリングは、フラットなグループとその下にあるレイヤをブレンドすることで得られます。
  - 子レイヤに混合モードが割り当てられている場合、フラット化の前に適用されますが、その範囲はそのグループの他の子レイヤにのみ影響し、グループ全体の下にある他のレイヤには影響

しないように制限されます。

- さらに **混合モード** のオプションがグループ内の子レイヤに用意されており、 **シンボロジ** タブでレンダリング中に他の子レイヤに対して「クリッピング」スタイルの操作を行うことができます。例えば、あるレイヤのレンダリング内容を、2 つ目の「マスク」レイヤの内容でクリップすることができます。

- **描画エフェクト**: *effects* を子レイヤのフラット化されたレンダリングにのみ適用します。例えば、グループに適用されたドロップシャドウのエフェクトは、隠蔽された子レイヤには見えません。

グループが **グループとしてレイヤを描画** に設定されている場合、そのグループのみが **レイヤ順序** パネルリストに表示されます。グループの子レイヤは、グループレイヤの配置によって順番が決まるので、この順番リストには表示されません。

レイヤスタイルを編集する

レイヤ パネルには、レイヤのレンダリングを素早く簡単に変更するショートカットがあります。

レイヤを右クリックし、リストから **スタイル** を選択します:

- レイヤで現在利用可能な **スタイル** が表示されます。レイヤにいくつもスタイルを定義している場合は、そのスタイルを切り替えることができ、マップキャンバス上のレイヤのレンダリングは自動的に更新されます。
- 現在のスタイルの一部または全部をコピーします。また、適用可能な場合には、別のレイヤからコピーしたスタイルを貼り付けます
- **現在のスタイル名を変更**...
- 新しいスタイルを **追加** （実際には現在のものをコピー）
- または :guilabel:`現在のスタイルを削除` （複数のスタイルが利用可能な場合のみ）。

Tip: レイヤスタイルを素早く共有

コンテキストメニューから、レイヤのスタイルをコピーし、レイヤのグループまたは選択範囲に貼り付けます: スタイルは、元のレイヤと同じ型（ベクタ、ラスタ、メッシュ、点群…）のすべてのレイヤで、ベクタレイヤの場合は同じジオメトリ型（ポイント、ライン、ポリゴン）を持つすべてのレイヤに適用されます。

地物の分類に基づくシンボロジを使用する場合（例えば、ベクタレイヤでは **カテゴリ値による定義**、 **連続値による定義**、 **ルールによる定義**、点群では **分類（*classification*）**）、 **レイヤ** パネルのクラスエントリを右クリックすると、分類（及びその地物）の可視性を編集できるようになり、1 つずつチェックする必要がなくなります:

- **アイテム切り替え**
- **全アイテム表示**
- **全アイテム非表示**

ベクタレイヤでは、クラスリーフエントリのコンテクストメニューからも次にアクセスできます：

- **地物を選択**: そのクラスにマッチしたすべての地物をレイヤで選択します
- **属性テーブルに表示**: そのクラスにマッチした地物でフィルタした属性テーブルを開きます
- **シンボルの色** を **カラーホイール** を使って更新します。 最近使用した色は、カラーホイールの下部からも選べて便利です。
- **シンボルを編集**...: 地物シンボル（シンボル、大きさ、色...）を変更する **シンボルセレクタ** ダイアログを開きます。
- **シンボルをコピーする**
- **シンボルを貼り付ける**

Tip: クラスリーフエントリをダブルクリックしても **シンボルセレクタ** ダイアログが開きます。

12.2.2 レイヤスタイル設定パネル

レイヤスタイル パネル（ `Ctrl+3` も可）は、 **レイヤプロパティ** ダイアログの機能の一部へのショートカットです。レイヤのレンダリングと動作を定義し、レイヤプロパティダイアログを開かずにその効果を視覚化するための迅速かつ簡単な方法を提供します。

レイヤスタイルパネルはレイヤプロパティダイアログによるブロッキング（または「モーダル」：ダイアログ操作以外は受け付けない状態）を回避することに加えて、ダイアログで画面が乱雑になることも避けられ、ほとんどのスタイル機能（カラーセレクタ、効果のプロパティ、ルール編集、ラベル置換...）が含まれています。例えば、レイヤスタイルパネル内の色ボタンをクリックすると、色セレクタダイアログが個別のダイアログとしてではなく、レイヤスタイルパネル自体の内部で開かれます。

レイヤパネルの現在のレイヤのドロップダウンリストから項目を選択して、以下の設定ができます：

- アクティブな項目に応じて、次を設定します:
  - グループの **シンボロジ** （**レイヤのレンダリングをグループで制御する** を参照）
  - ラスタレイヤの **シンボロジ** 、 **透過性** と **ヒストグラム** プロパティ。これらのオプションは **ラスタプロパティダイアログ** と同じものです。
  - ベクタレイヤの **シンボロジ** 、 **ラベル** 、 **マスク** 、 *3D* **ビュー** プロパティ。これらのオプションは **ベクタプロパティダイアログ** と同一のもので、サードパーティ製のプラグインによって導入されたカスタムプロパティによって拡張することができます。
  - メッシュレイヤの **シンボロジ** 、 *3D* **ビュー** プロパティ。これらのオプションは **メッシュデータセットのプロパティ** と同じものです。
  - 点群レイヤの **シンボロジ**、 *3D* **ビュー** 及び **標高** プロパティ。これらのオプションは **点群のプロパティ** と同じものです。

- 関連付けられたスタイルを  **スタイルをマネージャ** で管理する（詳細は **カスタムスタイルを管理する** を参照）。
- 現在のプロジェクトにおいてレイヤスタイル対して適用した変更の  **履歴** を確認する：リストの中から任意の状態を選択して **適用** をクリックすることで、レイヤスタイルの状態を戻したり復元させたりすることができます。

このパネルのもう 1 つの強力な機能は、  **ライブ更新** チェックボックスです。これにチェックを入れるとスタイル変更がマップキャンバスにすぐに表示され、 **適用** ボタンをクリックする必要はなくなります。

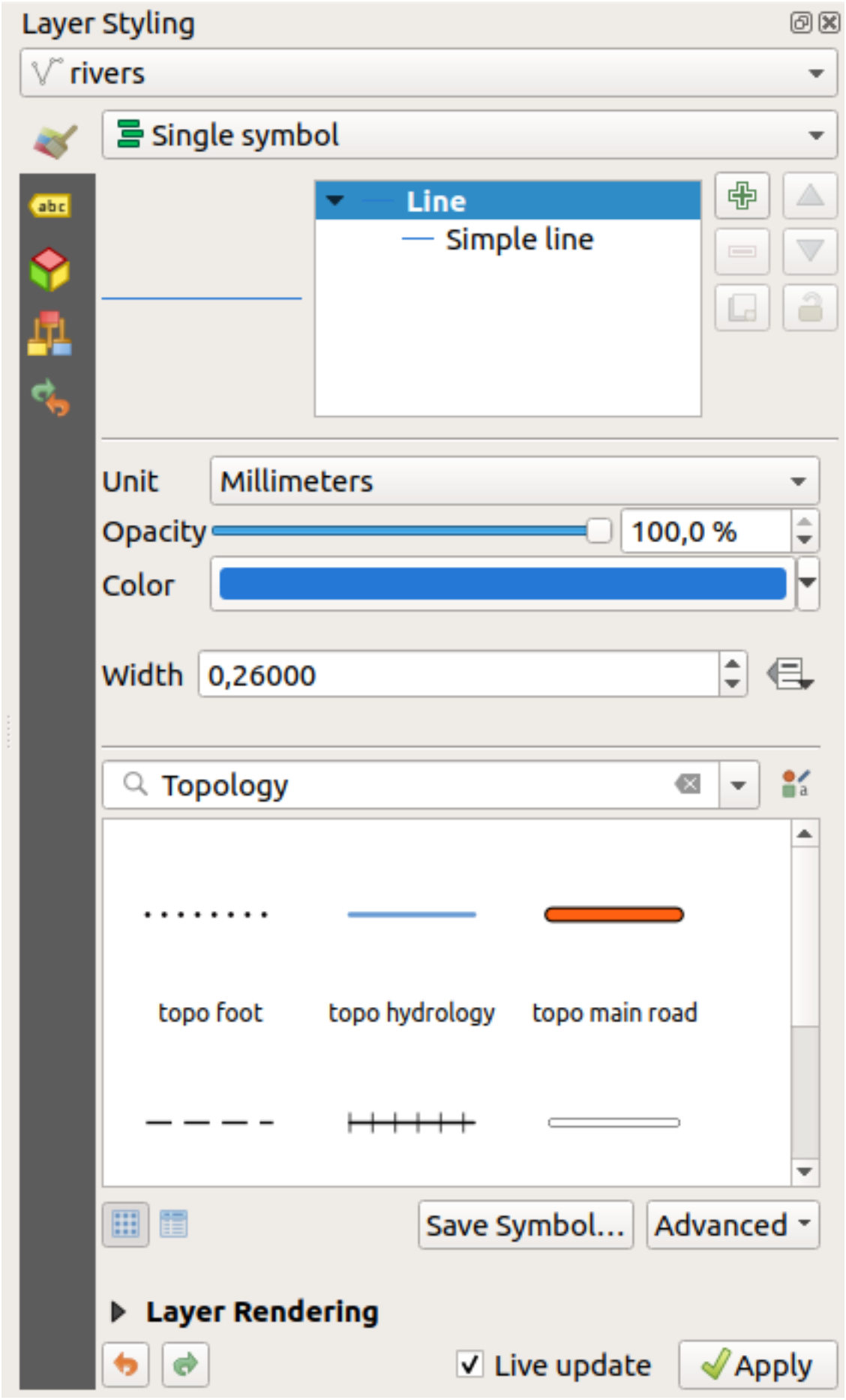


図 12.3: レイヤスタイルパネルからレイヤのシンボロジを定義する

12.2.3 レイヤ順序パネル

デフォルトでは、QGIS のマップキャンバスに表示されるレイヤは **レイヤ** パネルの順序に従って描画されます。つまりパネル内でレイヤの位置が高いほど、マップビューではより前面に（従って、より見えやすく）表示されます。

レイヤ順序 パネルを **ビュー** → **パネル** メニュー内からもしくは Ctrl+9 を押して有効にすると、レイヤパネル内の順序とは無関係にレイヤの描画順序を定義することができます。レイヤのリストの下にある **描画順序の制御** にチェックを入れて、パネル内のレイヤを好きなように再編成してください。この順序がマップキャンバスに適用される順序になります。例えば 図 12.4 の場合、レイヤパネル内でのそれぞれのレイヤ配置とは無関係に、「airports」 地物が「alaska」 ポリゴンの上に表示されていることがわかります。

描画順序の制御 のチェックを外すと、デフォルトの動作に戻ります。

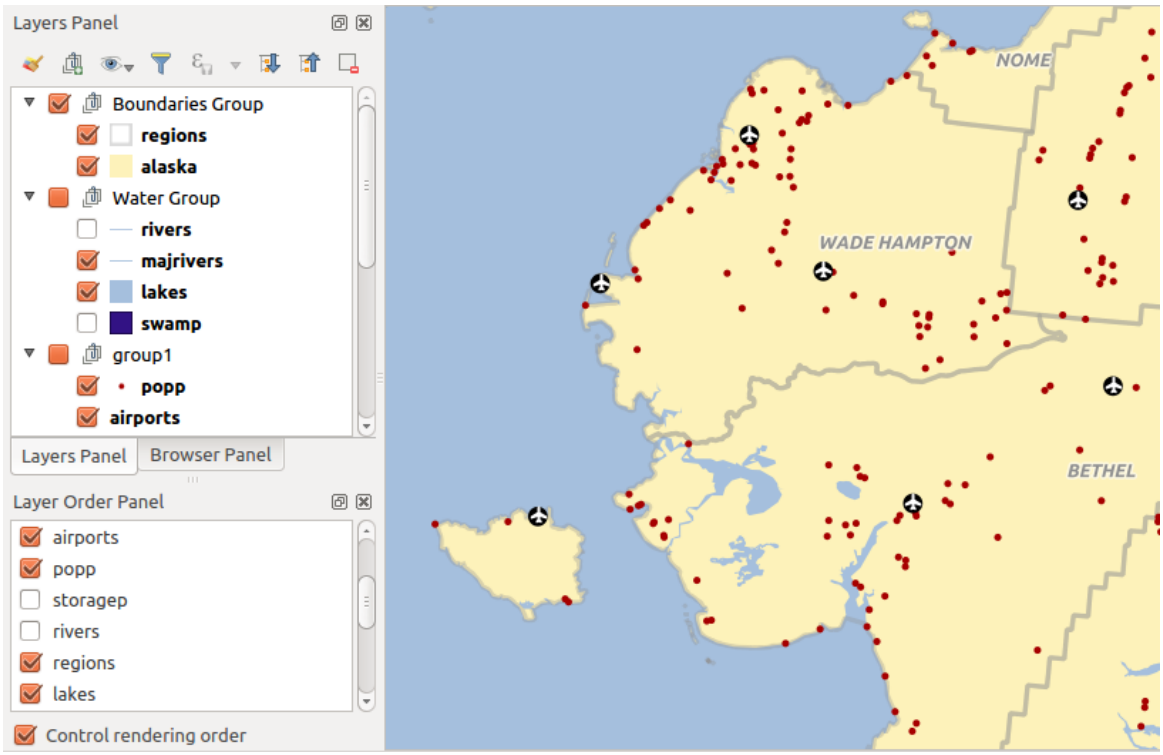


図 12.4: 凡例とは無関係にレイヤの順序を定義する

12.2.4 全体図パネル

全体図 パネル（ Ctrl+8 ）は、いくつかのレイヤの全範囲ビューの地図を表示します。全体図マップは、**レイヤ** メニューまたはレイヤのコンテキストメニューの **全体図に表示** オプションを使用するレイヤが表示されます。ビュー内では、現在のマップキャンバス範囲が赤い長方形で表示され、地図全体のどの領域が現在表示されているかをすばやく判断するのに役立ちます。全体図フレーム内の赤い長方形をクリックしてドラッグすると、それに応じてメインマップビューの範囲が更新されます。

全体図で使用するレイヤにラベル付けが設定されていても、全体図にラベルはレンダリングされないことに注意してください。

12.2.5 ログメッセージパネル

ロード時や、何らかの操作を処理するときは、 **ログメッセージパネル** を使用して別のタブに表示されるメッセージを追跡し従うことができます。これは、下のステータスバーの最右側のアイコンを使用して有効にできます。

12.2.6 元に戻す/やり直すパネル

元に戻す/やり直す （ Ctrl+5 ）パネルには、編集されたレイヤごとに実行されたアクションの一覧が表示されます。リストの上に表示されたアクションを選択することで、一連のアクションを素早く元に戻すことができます。さらなる詳細については、 **元に戻すとやり直す** を参照してください。

12.2.7 統計量の出力パネル

統計量の出力 パネル（ Ctrl+6 ）は、任意のベクタレイヤに関する情報の要約を提供します。このパネルでは、以下の事柄についての選択操作が可能です：

- 統計量を計算するベクタレイヤ: 一番上のドロップダウンメニューから選択するか、統計量ドロップダウンリストの一番下にある **選択した地物のみ** チェックボックスを使用して **レイヤ** パネル内のアクティブレイヤと同期させることができます
- 使用するフィールドまたは **式**: 各レイヤについて、最後のエントリーが記憶され、レイヤの再選択時に自動的に計算されます。
- 返される統計量はダイアログ右下にあるドロップダウンボタンを使って選択できます。フィールド（あるいは式の値）の型に応じて、利用可能な統計量は以下のとおりです：

表 12.2: 各フィールド型で利用できる統計量

統計	文字列	整数	浮動小数	日付
個数	☑	☑	☑	☑
個数（ユニークな値）	☑			☑
個数（欠損値）	☑	☑	☑	☑
合計（Sum ）		☑	☑	
平均値（Mean）		☑	☑	☑
標準偏差（母集団）		☑	☑	
標準偏差（標本）		☑	☑	
最小値	☑	☑	☑	☑
最大	☑	☑	☑	☑
範囲（Range）		☑	☑	☑
最稀値（Minority）	☑	☑	☑	
最頻値（Majority）	☑	☑	☑	
種類（Variety）		☑	☑	
第 1 四分位（Q1）		☑	☑	
第 3 四分位（Q3）		☑	☑	
四分位範囲（IQR）		☑	☑	
最短長（Min Length）	☑			
最大長（Max Length）	☑			
平均長（Mean Length）	☑			

統計量の概要は以下の設定・操作が可能です：

- レイヤ全体に対する統計量を返す、または ☑ **選択した地物のみ** にチェックを入れて選択した地物のみの統計量を返す
-  ボタンを押して統計量をクリップボードにコピーし、別のアプリケーションにテーブルとして貼りつけられるようにする
-  ボタンを押して、基礎となるデータソースに変更（例えば地物/フィールドの追加・削除や、属性値の変更）があった場合に統計量の再計算を行う

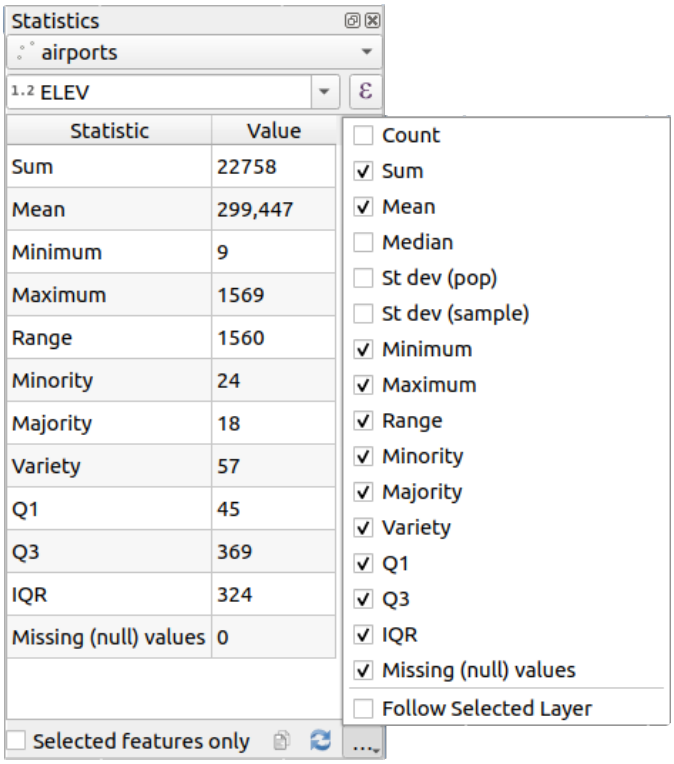


図 12.5: フィールドに対する統計量の一覧

12.2.8 デバッグ開発ツールパネル

デバッグ開発ツール パネル（F12）は、QGIS 内での対処やデバッグのための統一された場所を提供します。利用可能なツールは次のタブに整理されています:

- **ネットワークロガー**
- **クエリロガー**
- **プロファイラ**

注釈: プラグインの製作者は、独自のプラグインをデバッグしたり開発したりするために、このパネルをカスタムタブで拡張することができます。これは `registerDevToolWidgetFactory` メソッドを使って行います。

ネットワークロガー

ネットワークロガー タブは、リクエストとリプライのステータス、ヘッダー、エラー、SSL 設定エラー、タイムアウト、キャッシュステータスなど、便利な詳細情報と共に、ネットワークリクエストを記録して表示するのに役立ちます。

上部のツールバーから次ができます:

- **ログを保存する**: ロギングを開始または停止します。
- **ログをクリア**: ログ履歴を消去します。
- **ログを保存**...: 最初に、ログが機微であり、注意して扱われるべきであるという大きな警告を表示し、その後ログを保存できるようにします。
- **設定** ドロップダウンメニューを押して、 **完了リクエストを表示** するか、 **タイムアウトを表示** するか、 **キャッシュにある応答を表示** するかを選択します。
- **キャッシュを無効化**: キャッシュを無効にし、すべてのリクエストが実行されるようにします。
- **リクエストをフィルタ** は、URL 文字列のサブセットまたはリクエストステータスに基づきます

リクエストを右クリックと、以下の操作ができます：

- *URL* **を開く** デフォルトのブラウザで URL を開きます
- *URL* **をコピー**
- *cURL* **としてコピー** ターミナルで使用できます
- *JSON* **としてコピー**: ツリーの値を json 文字列としてクリップボードにコピーし、バグレポートやリモートアシスタンスに簡単に貼り付けられるようにします。

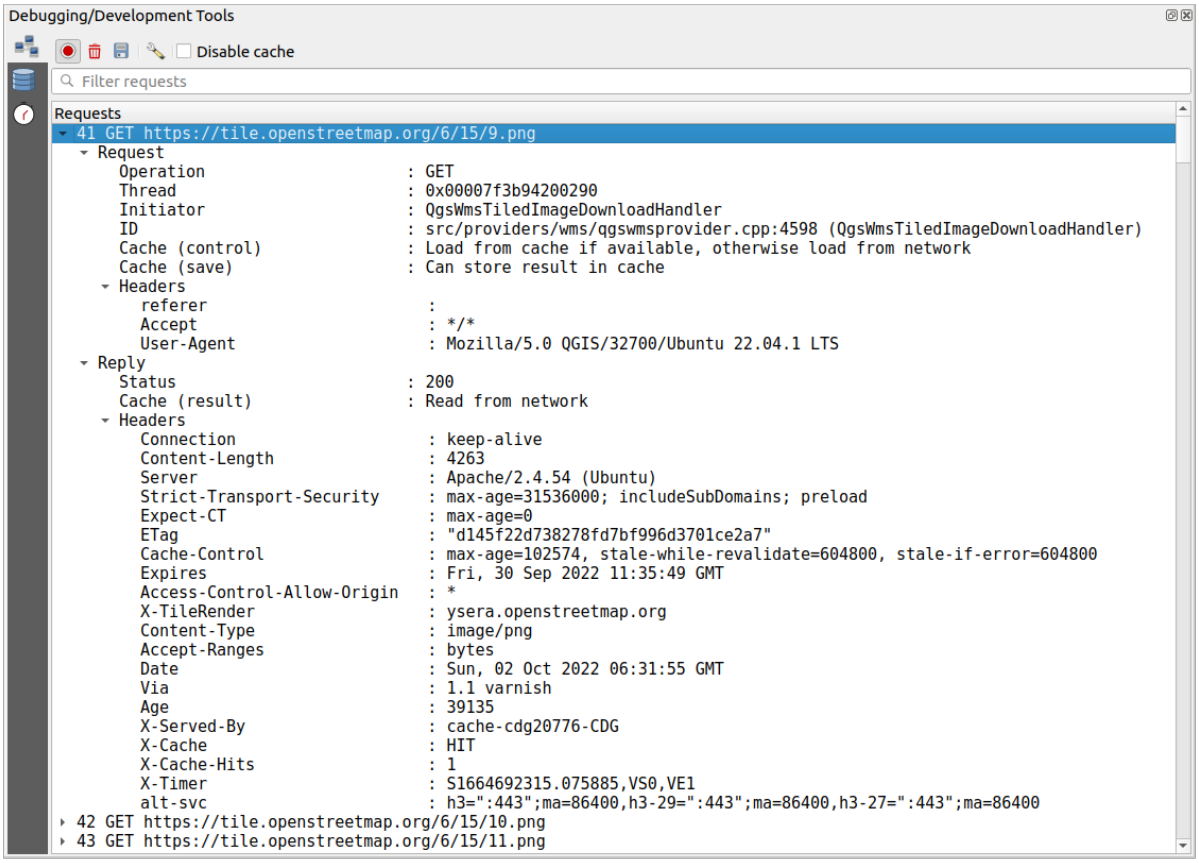


図 12.6: GET リクエストのネットワークロガー出力

クエリロガー

クエリロガー は、データプロバイダと接続 API からバックエンドデータベースに送信された SQL コマンドと、QGIS で計測された（つまりコマンドを送信したクライアントでの）実行時間を記録する場所です。これは、QGIS アルゴリズムやプラグインのデバッグや開発中に特定のレイヤーのパフォーマンスを調査する際に役立ちます。

上部のツールバーから次ができます:

- **ログを保存する**: ロギングを開始または停止します。
- **ログをクリア**: ログ履歴を消去します。
- **ログを保存**...: 最初に、ログが機微であり、注意して扱われるべきであるという大きな警告を表示し、その後ログを保存できるようにします。
- **クエリをフィルタ** プロバイダタイプ、開始時間、イニシエータ... などのクエリ文字列のサブセットや詳細に基づきます

報告されたクエリで右クリックすると、次ができます:

- *SQL* **をコピー** は、QGIS からデータベースに発行されたコマンドをコピーします

- *JSON* **としてコピー**: ツリーの値を json 文字列としてクリップボードにコピーし、バグレポートやリモートアシスタンスに簡単に貼り付けられるようにします。

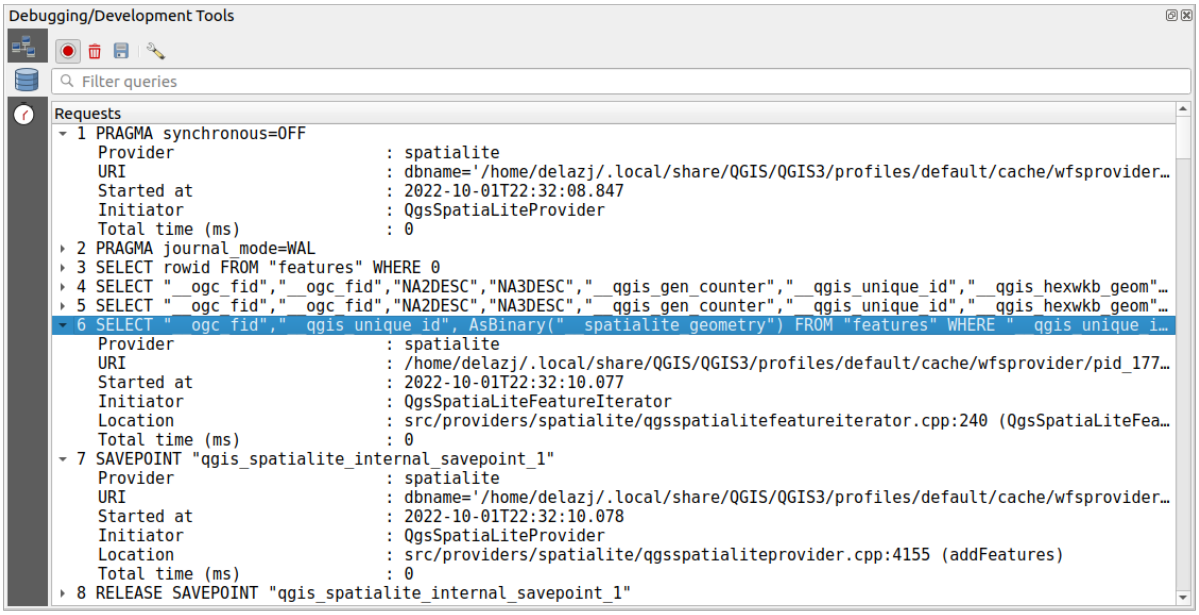


図 12.7: クエリロガーの出力

プロファイラ

プロファイル タブを使用すると、ユーザーによって要求されたアクションに関わるあらゆる操作の読み込み時間を取得することができます。コンテキストによって、これらの操作は設定の読み込み、メニュー、マップキャンバスや 3D ビューの作成、マップレイヤの参照解決、ブックマークやレイアウトの読み込みなどがあります。これは、読み込み時間が遅い原因を特定するのに役立ちます。

デフォルトでサポートされているアクションは **カテゴリ値の出力** ドロップダウンメニューから選択できます:

- QGIS の **起動**
- **プロジェクトの読み込み**

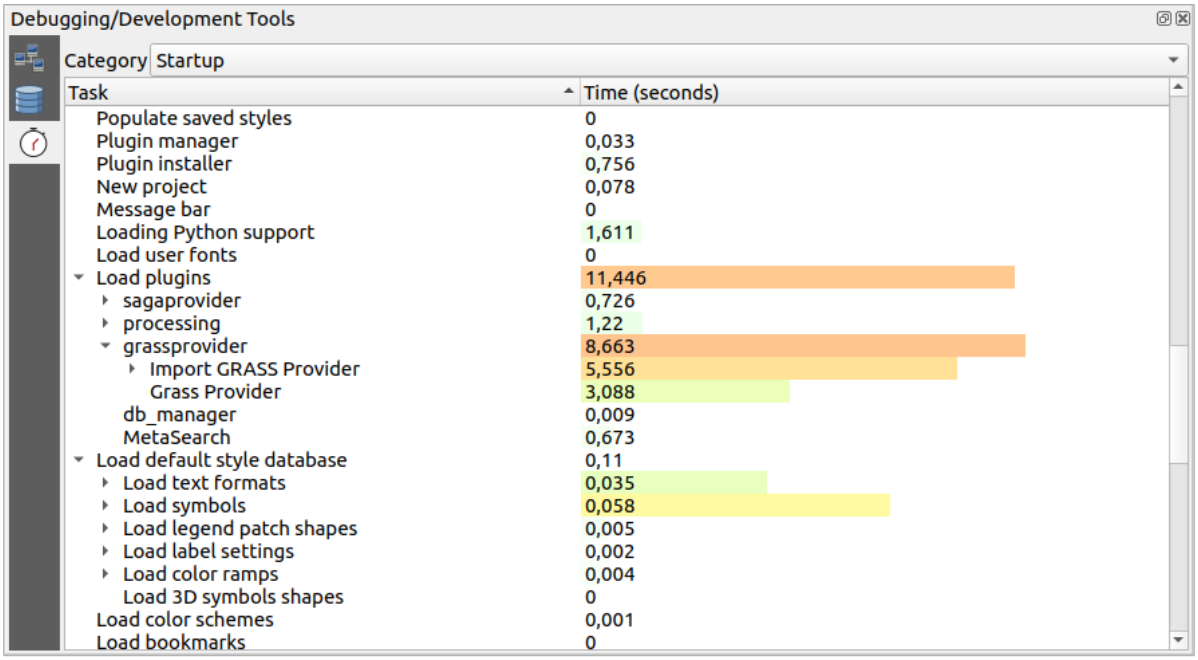


図 12.8: QGIS の起動に関するプロファイラ

12.3 外部プロジェクトからのレイヤの埋め込み

時には、さまざまなプロジェクトにある一部のレイヤについて、同じスタイルでレイヤを保持したい場合もあるでしょう。これらのレイヤに対して **デフォルトスタイル** を作成するか、別のプロジェクトからそのレイヤを埋め込むことで、時間と労力を節約することができます。

既存のプロジェクトからレイヤやグループを埋め込むことには、スタイル設定を超える利点がいくつかあります：

- あらゆる種類のレイヤ（ベクタ/ラスタ、ローカル/オンラインなど...）を追加できます
- グループとレイヤを取得することで、異なるプロジェクトで「背景」レイヤのツリー構造を同じに保つことができます
- 埋め込まれたレイヤは編集可能ですが、シンボロジ、ラベル、フォーム、デフォルト値やアクションなどのプロパティを変更することはできないため、プロジェクト間で一貫性を確保できます
- 元のプロジェクト内の項目を変更すると、変更は他のすべてのプロジェクトにも反映されます。

他のプロジェクトのコンテンツをあるプロジェクトに埋め込みたい場合には、 **レイヤ　レイヤとグループを埋め込む** を選択して、以下のように操作します：

1. ... ボタンをクリックし、プロジェクトを探します。プロジェクトの中身を確認することもできます（ 図 12.9 を参照）
2. `Ctrl` キー（ の場合は `Cmd` キー）を押しながら取得したいレイヤやグループをクリックします
3. *OK* をクリックします

選択したレイヤとグループは **レイヤ** パネルに埋め込まれ、マップキャンバスに表示されます。識別のために アイコンが名前の横に追加され、この上にマウスカーソルを置くと、元のプロジェクトファイルパスを示すツールチップが表示されます。

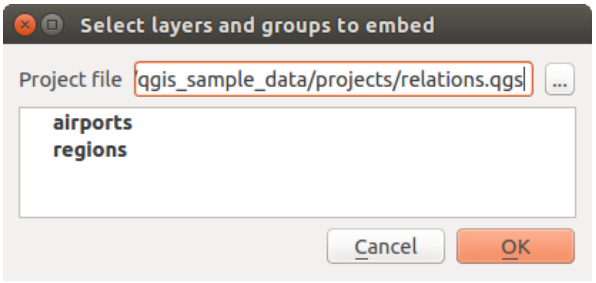


図 12.9: 埋め込むレイヤとグループの選択

他のレイヤと同じように、埋め込まれたレイヤもレイヤを右クリックして レイヤの削除 をクリックすることで、プロジェクトから削除することができます。

Tip: 埋め込みレイヤのレンダリングを変更する

元のプロジェクトファイルに変更を加えない限り、埋め込みレイヤのレンダリングは変更できません。しかし、レイヤ上で右クリックして **複製** を選択すると、完全な機能を備えた、元のプロジェクトに依存していないレイヤが作成されます。その後は、リンクされた埋め込みレイヤを削除しても問題ありません。

12.4 地物とのやりとり

12.4.1 地物の選択

QGIS には、マップキャンバス上の地物を選択するためのツールがいくつかあります。選択ツールは、 **編集** **選択** メニューや **選択ツールバー** にあります。

注釈: 選択ツールは、現在アクティブなレイヤで動作します。

マップキャンバス上の手動選択

マウスを使って 1 つもしくは多数の地物を選択するには、以下のツールのいずれかを使います：

- エリアまたはシングルクリックによる地物選択
- ポリゴンによる地物選択
- フリーハンドによる地物選択
- 円による地物選択

注釈:  ポリゴンによる地物選択 を除き、これらの手動選択ツールはマップキャンバス上の地物をシングルクリックで選択することもできます。

注釈: （任意レイヤの）既存のポリゴン地物を使用して、アクティブレイヤ内で重なる地物を選択するには  ポリゴンによる地物選択 ツールを使用してください。ポリゴンを右クリックして、クリックした点を含むすべてのポリゴンのリストを表示するコンテキストメニューから、使用したいポリゴンを選択します。すると、ポリゴンと重なり合うアクティブレイヤのすべての地物が選択されます。

Tip: 編集　選択　地物を再選択 ツールを使うと、直前に行った選択をやり直せます。これは苦労して選択をした後、うっかりどこか別の場所をクリックしてしまい、選択がクリアされてしまった時に非常に便利です。

 **地物の選択** ツールを使用中に `Shift` キーや `Ctrl` キーを押しながら選択すると、地物の選択状態が反転します（つまり、現在の選択に追加するか、選択から除くかをします）。

他のツールについては、以下のキーを押しながら選択した際の動作が異なります：

- `Shift` ：地物を現在の選択に追加します
- `Ctrl` ：地物を現在の選択から除きます
- `Ctrl+Shift`: 現在の選択範囲との交差を取ります。すなわち、現在の選択範囲と重なりがある地物のみを残します
- `Alt`: 選択形状の中に完全に入っている地物を選択します。 `Shift` キーや `Ctrl` キーと組み合わせると、地物を現在の選択に追加したり、選択から引いたりできます。

自動選択

その他の選択ツールは、地物の属性やその選択状態に基づいて選択を実行します。ツールのほとんどは **属性テーブル** からも利用できます（属性テーブルと地図のキャンバスは同じ情報を表示するので、属性テーブル内で 1 つの地物を選択した場合、それはまた、地図キャンバスでも選択されることに注意してください）。

-  式による地物選択... 式ダイアログを使用して地物を選択します
-  値による地物選択... または `F3` を押します
-  全レイヤの選択を解除 または `Ctrl+Alt+A` を押しても全レイヤの地物選択を解除することができます
-  アクティブレイヤの選択を解除 または `Ctrl+Shift+A` を押します
-  すべての地物を選択 または `Ctrl+A` でも現在のレイヤの地物全てを選択します
-  地物選択を反転 現在のレイヤ内の地物の選択状態を反転させます

-  場所による選択 は、他の地物（同じレイヤもしくは他のレイヤ - **場所による選択** 参照）との空間的な関係に基づいて地物を選択します

例えば、QGIS サンプルデータの `regions.shp` から Borough（市）の地域を探したい場合には、次のようにします：

1.  式による地物選択 アイコンをクリックします
2. **フィールドと値** グループを展開します
3. クエリしたいフィールド（ "TYPE_2" ）をダブルクリックします
4. 右に現れたパネルの **全てのユニーク** ボタンをクリックします
5. リストの中から、「Borough」（市）をダブルクリックします。 **式** エディタフィールドで、以下のようにクエリを記述します：

   ```
   "TYPE_2"  =  'Borough'
   ```

6. **地物の選択** ボタンをクリックします

式ビルダーダイアログから、 **関数リスト**　**最近使用（*selection*）** を使用して、以前に使用した式による選択を行うこともできます。ダイアログには最後に使用された 20 個の式が記憶されます。詳細情報や例は **式** を参照してください。

Tip: 新しいファイルに選択地物を保存する

編集　**地物のコピー** と **編集**　**新規レイヤへの地物貼り付け** を使用することで、ユーザは選択した地物を **一時スクラッチレイヤ** や任意のファイル形式の **新規ベクタレイヤ** に保存することができます。

値による地物選択

この選択ツールはレイヤの地物フォームを開き、各フィールドで検索する値、大文字と小文字を区別するかどうか、使用する操作をユーザーが選択できるようにします。このツールには自動補完機能もあり、自動的に検索ボックスに既存の値を入力します。

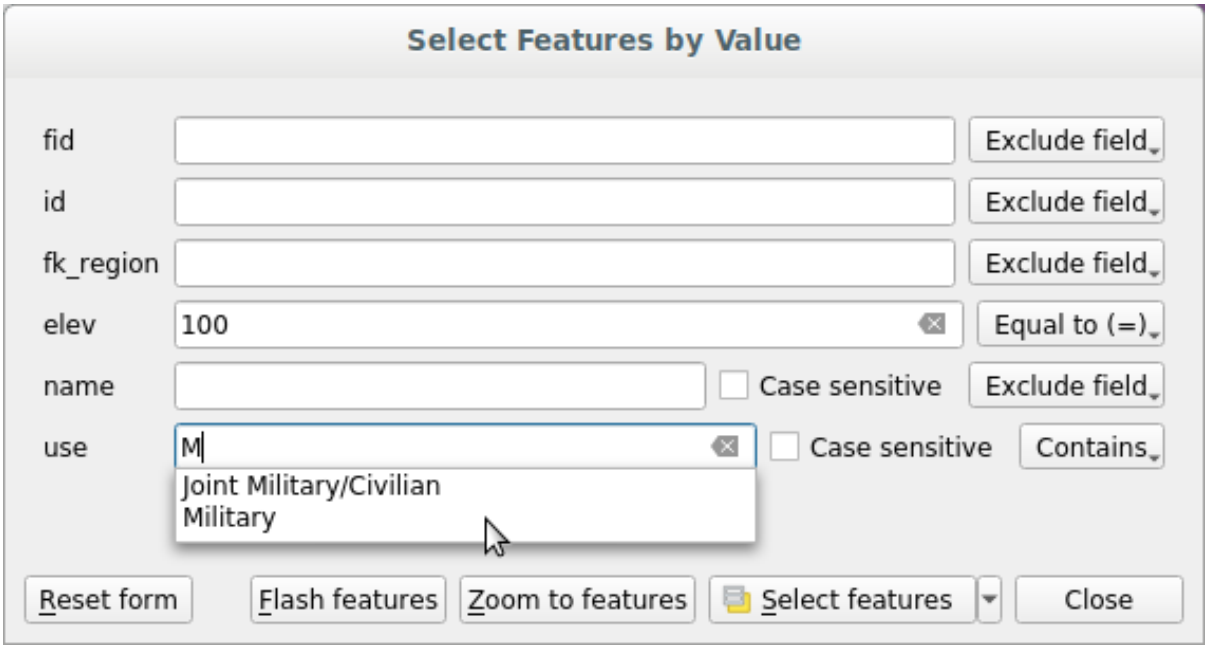


図 12.10: フォームダイアログを使用して地物をフィルタ/選択

各フィールドの横には、検索動作を制御するためのオプションを選べるドロップダウンリストがあります：

表 12.3: データ型毎のクエリ操作

フィールド検索オプション	文字列	数値	日付
フィールドを除外 ：検索対象からフィールドを除外します	☑	☑	☑
等しい（=）	☑	☑	☑
等しくない（≠）	☑	☑	☑
より大きい（>）		☑	☑
より小さい（<）		☑	☑
以上（≧）		☑	☑
以下（≦）		☑	☑
範囲内（境界値を含む）		☑	☑
範囲外（境界値を含む）		☑	☑
含む	☑		
含まない	☑		
値がない（*null*）	☑	☑	☑
値がある（*null* ではない）	☑	☑	☑
で始まる	☑		
で終わる	☑		

文字列の比較には、☑ *Case sensitive* オプション（大文字・小文字を区別する）を使用することもできます。

全ての検索オプションを設定し終えたら、 **地物を選択** ボタンをクリックしてマッチする地物を選択します。ドロップダウンオプションには以下のものがあります：

- **地物を選択**
- **現在の選択に追加する**
- **現在の選択から除去する**
- **現在の選択をフィルタする**

フォームのリセット ボタンを押すことで、全ての検索オプションをクリアすることができます。

検索条件をセットしたら、以下の操作も行えます：

- **地物にズーム** あらかじめ選択することなく、マップキャンバス上の地物にズームします
- **地物をフラッシュ** 検索条件に一致する地物を強調表示します。これは、選択や地物情報表示ツールを使用せずに地物を識別するのに便利な方法です。「地物をフラッシュ」はマップキャンバスの範囲を変更しないため、地物が現在の地図キャンバスの範囲内にある場合にのみ強調表示されることに注意してください。

12.4.2 地物の識別

地物情報表示ツールを使用すると、地図キャンバスと対話的に操作して地物に関する情報を取得し、ポップアップウィンドウ内に表示します。地物情報を表示するには、以下を使用します：

- **ビュー** **地物情報表示**
- `Ctrl+Shift+I` （ の場合は `Cmd+Shift+I` ）
- 属性ツールバーの 地物情報表示 アイコン

地物情報表示ツールを使用する

QGIS には 地物情報表示 ツールを使用して地物の情報を表示する方法がいくつかあります：

- **左クリック 地物情報** パネルの **選択モード** と **選択マスク** の設定に応じて、地物の情報を表示します
- **右クリック 地物情報** パネルで **選択モード** 設定として **地物情報表示** ツールで右クリックすると、全ての可視レイヤからスナップされた地物全てを取得します。コンテキストメニューが開かれ、ユーザは情報表示しようとする地物をより正確に選択したり、あるいは地物に対して実行するアクションを選択したりすることができます。
- **右クリック 地物情報** パネルで **選択モード** を **ポリゴンによる地物の特定** とした状態で右クリックすると、 **地物情報** パネルの **選択マスク** 設定に応じて、選択した既存のポリゴンと重複する地物の情報を表示します。

Tip: 地物情報表示ツールでクエリするレイヤをフィルタリングする

プロジェクト **プロパティ...** **データソース** タブ内の **レイヤの** *Capabilities* で、レイヤ列の隣にある **情報表示可能** 列のチェックを外すと、 **カレントレイヤ** モード以外のモードで 地物情報表地 ツールを使った

場合に地物情報が検索されなくなります。これは、興味のあるレイヤのみから地物情報を返すための便利な方法です。

地物をクリックすると、**地物情報** ダイアログにクリックされた地物に関する情報が一覧表示されます。 デフォルトビューはツリービューで、最初の項目はレイヤの名前、その子は識別された地物です。 各地物は、フィールドの名前および値で記述されます。 このフィールドは **レイヤプロパティ　　表示名** で設定されたものです。 地物に関する他のすべての情報が続きます。

地物情報

地物情報ダイアログはカスタムのフィールドを表示するようにカスタマイズすることもできますが、デフォルトでは以下の情報を表示します：

- 地物の **表示名**
- **アクション**：アクションを地物情報ウィンドウに追加できます。アクションは、アクションのラベルをクリックすることで実行されます。デフォルトでは、編集のため 地物フォームを見る のアクション 1 つのみが追加されています。レイヤプロパティダイアログでより多くのアクションを定義することができます（ **アクションプロパティ** を参照）。
- **派生した属性** ： この情報は、他の情報から計算されたものや、派生したものです。これには以下のものがあります：
  - 地物のジオメトリに関する一般的な情報：
    - ジオメトリのタイプにもよるが、レイヤの CRS の単位で表示したデカルト計測による長さ、周長、面積。3 次元のラインベクタレイヤでは、ラインのデカルト長さも利用可能です。
    - ジオメトリのタイプにもよるが、プロジェクトのプロパティダイアログにおいて **計測** に楕円体が設定されている場合には、指定された単位で表示した楕円体計算による長さ、周長、面積
    - 地物内のジオメトリパーツの数と、クリックされたパーツの番号
    - 地物内の頂点数
  - プロジェクトのプロパティ **座標の表示** 設定を使用した座標情報：
    - クリックされた点の `X` 、 `Y` 座標値
    - クリックされた点に最も近い頂点の番号
    - 最も近い頂点の `X` 、 `Y` 座標値（利用可能ならば `Z` / `M` も）
    - 曲線セグメントをクリックした場合には、その部分の曲率半径も表示されます。
- **データ属性** ：これは、クリックされた地物の属性フィールドと値のリストです。
- **リレーション** を定義している場合には、関連する子地物についての情報
  - リレーションの名前
  - 参照フィールドのエントリ、例えば関連した子地物の名前

- **アクション** ：レイヤプロパティダイアログで定義されているアクション（ **アクションプロパティ** 参照）をリストします。デフォルトのアクションは 地物フォームを見る です。
- **データ属性** ：これは、関連する子地物の属性フィールドと値のリストです。

注釈: 地物属性内のリンクは **地物情報** パネルでクリックすることができ、デフォルトのウェブブラウザでリンクを開きます。

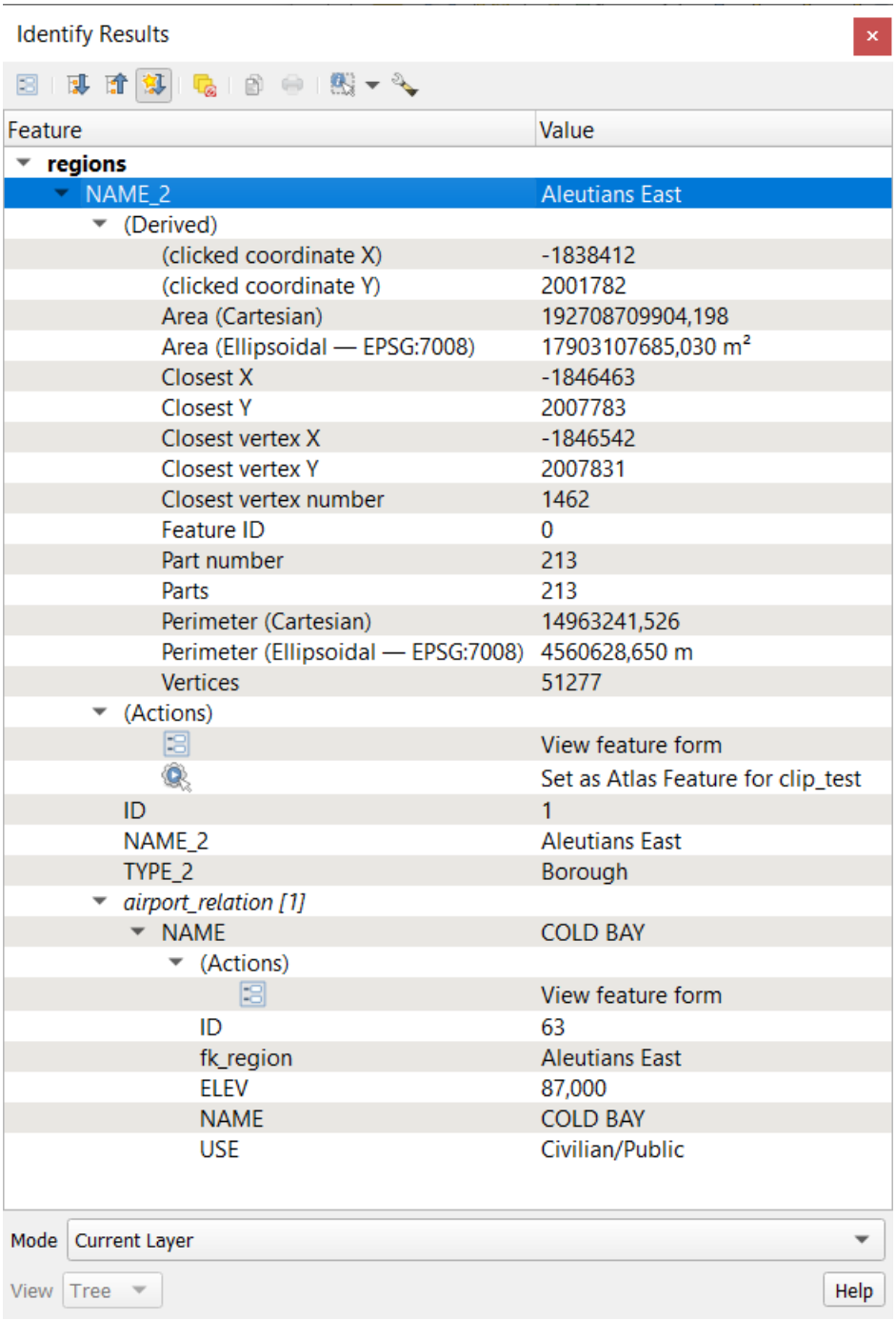


図 12.11: 地物情報ダイアログ

地物情報ダイアログ

ウィンドウの上部には、少数のツールがあります：

-  地物フォームを見る 現在の地物の属性フォームを開きます
-  ツリーを展開する
-  ツリーを折りたたむ
-  結果をデフォルトで展開する 次に識別する地物情報のツリーを折りたたむか、展開するかを規定します
-  結果のクリア
-  選択地物をクリップボードにコピー
-  選択した HTML レスポンスを印刷
- 情報表示したい地物を取得するための選択モードには、以下のものがあります：
  -  地物を特定する
  -  ポリゴンによる地物の特定
  -  フリーハンドで地物を識別する
  -  半径による地物の特定

注釈:  ポリゴンによる地物の特定 を使用した際には、任意の既存のポリゴンを右クリックし、このポリゴンを重なりのある他のレイヤ内の地物を特定するために使用することができます。

ウィンドウの下部には、 **モード** と **ビュー** のコンボボックスがあります。 **モード** は、どのレイヤの地物情報を表示するかを定義します：

- **現在のレイヤ** ：選択したレイヤの地物情報のみが表示されます。レイヤのグループを選択した場合には、そのグループの可視レイヤの地物が識別されます。選択レイヤが無い場合には、現在のレイヤのみが識別されます。
- **トップダウン 最初の結果のみ** ： 最も上に見えているレイヤの地物のみを情報表示します。
- **トップダウン** ：全ての可視状態レイヤの地物情報を表示します。結果はパネル内に表示されます。
- **レイヤ選択** ：右クリック時と同様に、コンテキストメニューを開き、どのレイヤの地物を情報表示するのかユーザがレイヤを選択します。パネルに表示されるのは選択されたレイヤの地物情報のみです。

ビュー は **ツリー** 、 **テーブル** または **グラフ** として設定できます。「テーブル」と「グラフ」ビューはラスタレイヤに対してのみ設定できます。

地物情報表示ツールでは  地物情報表示の設定 から  **単一地物の場合、自動でフォームを開く** の設定が行えます。これにチェックを入れると、単一の地物の情報を表示する場合には、フォームが開き地物属性が表示されます。これは地物の属性情報を素早く編集する際に便利な方法です。

その他の機能は、識別されたアイテムのコンテキストメニューにあります。たとえば、コンテキストメニューから次のことができます：

- 地物フォームを表示
- 地物にズーム
- 地物をコピー: すべての地物ジオメトリと属性をコピーします
- 選択地物を切り替え：識別された地物を選択範囲に追加します
- 属性値のコピー: クリックした属性値のみをコピーします
- 地物属性のコピー：地物属性をコピーします
- 属性で地物を選択：レイヤにある選択した属性にマッチするすべての地物を選択します
- 結果のクリア：ウィンドウ内の結果が削除されます
- ハイライトをクリア：マップ上の地物ハイライトが除去されます
- すべてをハイライト
- レイヤを強調表示
- レイヤをアクティブにする：アクティブにするレイヤを選択します
- レイヤのプロパティ: レイヤプロパティウィンドウを開きます
- すべて展開する
- すべて折りたたむ

12.5 レイヤのプロパティの保存および共有

12.5.1 カスタムスタイルを管理する

ベクタレイヤがマップキャンバスに追加されると、QGIS はデフォルトでランダムなシンボル/色を使用して地物をレンダリングします。ただし、 **プロジェクト** **プロパティ...** **既定スタイル** でデフォルトのシンボルを設定することができ、新しく追加された各レイヤには、ジオメトリタイプに応じてこれが適用されます。

しかし、ほとんどの場合、レイヤに自動的にまたは手動で（より少ない労力で）適用できる、カスタマイズされたより複雑なスタイルが欲しいと思うでしょう。これはレイヤのプロパティダイアログの下部にある **スタイル** メニューを使えば実現できます。 このメニューには、スタイルを作成、読み込み、管理するための機能があります。

スタイルは、ベクタレイヤの場合はレイヤのレンダリングや相互作用（シンボロジ、ラベル付け、フィールドやフォームの定義、アクション、ダイアグラム...)、ラスタレイヤの場合はピクセル（バンド、色レンダリング、透明度、ピラミッド、ヒストグラム...）に関する、レイヤプロパティダイアログで設定された任意の情報を保存します。

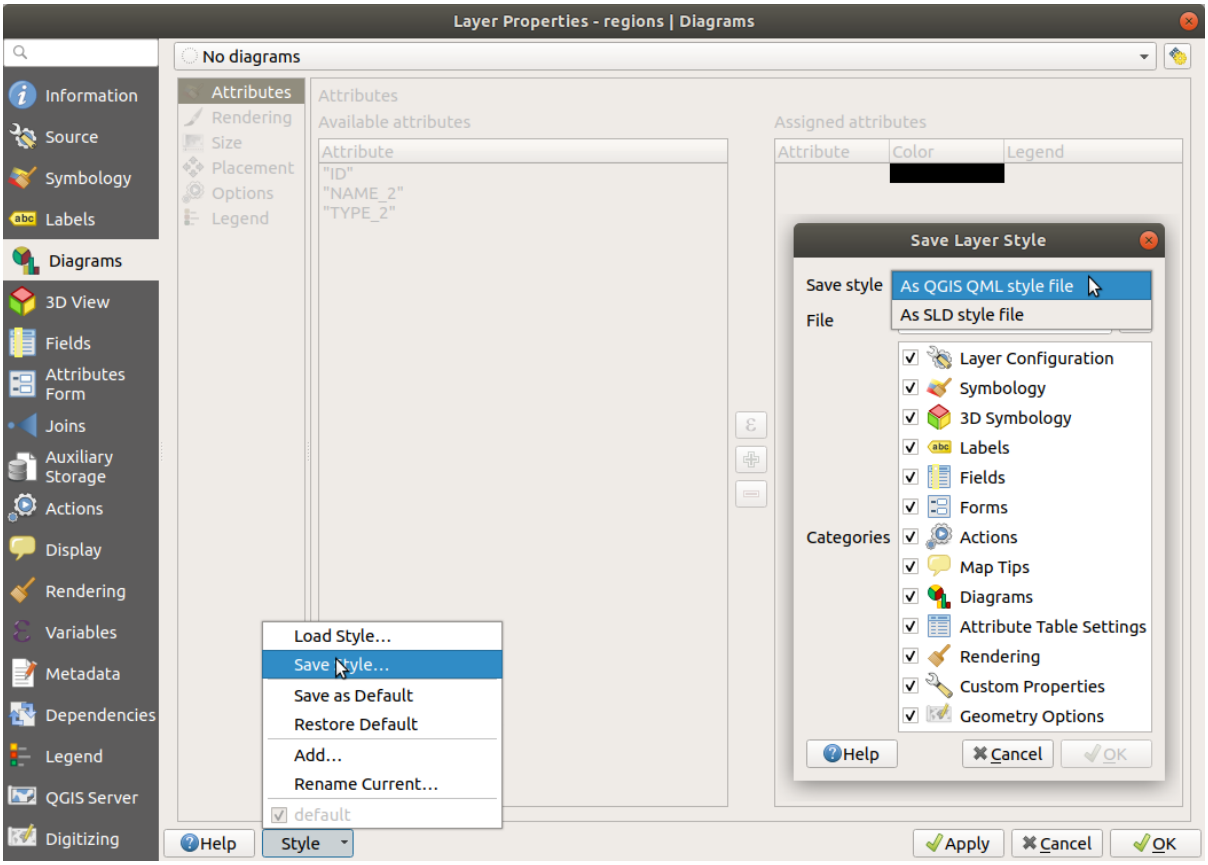


図 12.12: ベクタレイヤのスタイルコンボボックスのオプション

デフォルトでは、読み込まれたレイヤに適用されるスタイルは `default` という名前です。あなたのレイヤに理想的で適切なレンダリングが設定できたら、 **スタイル** コンボボックスをクリックして下記の項目を選択することで、スタイルを保存することができます。

- **現在のスタイル名を変更** ：現在アクティブなスタイルは名前が変更され、現在のオプションで上書きされます
- **追加** ：現在のオプションを使用して新しいスタイルが作成されます。デフォルトでは、QGIS プロジェクトファイルに保存されます。別のファイルやデータベースにスタイルを保存するには、以下を参照してください。
- **現在のスタイルを削除** ：レイヤに複数のスタイルが定義されている場合は、不要なスタイルを削除します。

スタイルのドロップダウンリストの最下部で、レイヤに設定されているスタイルのうちアクティブなものにチェックが入っていることを確認できます。

レイヤのプロパティダイアログでスタイルの適用を確定させるたびに、アクティブなスタイルは変更された内容で更新されることに注意してください。

あるレイヤに対するスタイルは、望むならば好きなだけ作成することができますが、一度にアクティブにできるのは 1 つだけです。 **地図のテーマ** と組み合わせることで、 マップ凡例のレイヤを複製することなく、複雑なプロジェクトを迅速かつ強力に管理することができます。

注釈: レイヤのプロパティに変更を適用すると、変更内容がアクティブなスタイルに保存されるため、**地図テーマ** で使用しているスタイルを誤って変更しないよう、正しいスタイルを編集していることを常に確認するようにしましょう。

Tip: レイヤのコンテキストメニューからスタイル管理

レイヤ パネル内のレイヤを右クリックして、レイヤのスタイルをコピー、貼り付け、追加、スタイル名変更ができます。

12.5.2 スタイルをファイルやデータベースに保存する

スタイル コンボボックスから作成されたスタイルは、デフォルトではプロジェクト内に保存され、プロジェクト内のレイヤ間でスタイルをコピー・貼りつけできますが、スタイルをプロジェクト外に保存し、別のプロジェクトで読み込めるようにすることもできます。

テキストファイルに保存

 **スタイル** **スタイルを保存** をクリックすることで、スタイルは以下の形式で保存できます：

- QGIS QML スタイルファイル（ `.qml` ）
- SLD スタイルファイル（ `.sld`)、ベクタレイヤのみ可能

ファイルベース形式のレイヤ（ `.shp` 、 `.tab`... ）を使っている場合、 **デフォルトとして保存** はそのレイヤのための（ファイルと同名の） `.qml` ファイルを生成します。SLD ファイルは任意のタイプのレンダラ（単一定義のシンボル、カテゴリ値による定義、連続値による定義、あるいは、ルールによる定義）からエクスポートすることができますが、SLD ファイルをインポートする際は、単一定義もしくはルールによる定義のレンダラが生成されます。これは、カテゴリ値もしくは連続値による定義のスタイルは、ルールによる定義のスタイルへと変換されることを意味します。これらのレンダラを維持したい場合には、QML 形式を使用する必要があります。一方で、このように簡単にスタイルをルールによる定義へと変換できる方法があると、非常に便利なこともあります。

データベースに保存

レイヤのデータソースがデータベースプロバイダの場合、ベクタレイヤスタイルをデータベースに保存することもできます。サポートされているフォーマットは PostGIS、 GeoPackage、SpatiaLite、MS SQL Server、Oracle です。レイヤスタイルはデータベース内の（ `layer_styles` という名前の）テーブルに保存されます。**保存 スタイル**... **データベースに保存** をクリックし、ダイアログでスタイル名の決定、説明の追加、 `.ui` ファイルの追加、デフォルトスタイルにするかどうかのチェックを行います。

データベースの単一テーブルに複数のスタイルを保存することもできます。しかしながら、各テーブルでデフォルトのスタイルは 1 つだけしか持つことができません。デフォルトのスタイルは、レイヤのデータ

ベースまたはアクティブな **ユーザープロファイル** ディレクトリにあるローカルな SQLite データベースである `qgis.db` 内に保存することができます。

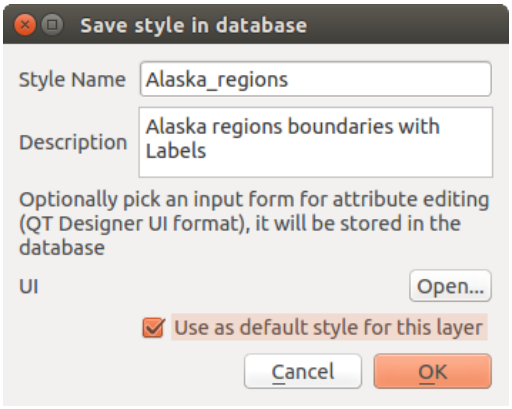


図 12.13: データベースにスタイルを保存ダイアログ

Tip: データベース間でスタイルファイルを共有する

スタイルをデータベースに保存できるのは、レイヤがそのようなデータベースに由来する場合のみです。（例えば、Oracle のレイヤと MS SQL Server のスタイルのように）データベースを混合させることはできません。データベース間でスタイルを共有したい場合は、代わりにプレーンテキストファイルを使用してください。

注釈: PostgreSQL データベースのバックアップから `layer_styles` テーブルを復元する際に問題が起こることがあります。この問題を修正するには、 *QGIS layer_style* **テーブルとデータベースのバックアップ** に従ってください。

スタイルの読み込み

QGIS にレイヤを読み込むとき、レイヤにデフォルトのスタイルが存在する場合には、QGIS はそのスタイルでレイヤを読み込みます。また、 **スタイル　デフォルトとして保存** はそのファイルを探して読み込み、レイヤの現在のスタイルでそのファイルを置き換えます。

スタイル　スタイルを読み込む で、任意の保存されたスタイルをレイヤに適用させることができます。テキスト形式のスタイル（ `.sld` や `.qml` ）はフォーマットが何であれどのようなレイヤにも適用できますが、データベース内に保存されたスタイルを読み込むのは、レイヤが同じデータベースのものであるか、またはスタイルが QGIS のローカルデータベースに保存されている場合にのみ可能です。

DB **マネージャ** ダイアログは、データベース内で見つかったレイヤに関連するスタイルや、データベース内に保存されているその他スタイルのリストを名前と説明付きで表示します。

Tip: プロジェクト内でのレイヤスタイルの素早い共有

ファイルやデータベーススタイルをインポートせずに、プロジェクト内でレイヤのスタイルを共有することもできます： **レイヤパネル** でレイヤを右クリックし、 **スタイル** コンボボックスからレイヤのスタイルをコピーして、レイヤのグループや選択したレイヤに貼りつけます。このスタイルは、元のレイヤと同じタイプ（ベクタ、ラスタ）で、ベクタレイヤの場合は同じジオメトリタイプ（ポイント、ライン、ポリゴン）を持つ全てのレイヤに適用されます。

12.5.3 レイヤ定義ファイル

レイヤ定義は、アクティブなレイヤのコンテキストメニューで **エクスポート　レイヤ定義ファイルとして保存**... を使用することで、 レイヤ定義ファイル（ `.qlr` ）として保存することができます。レイヤ定義ファイル（ `.qlr` ）には、 レイヤのデータソースとスタイルへの参照が含まれています。 `.qlr` ファイルはブラウザパネルに表示され、レイヤパネルにレイヤ（と保存されたスタイル）を追加するために使用できます。システムのファイルマネージャからマップキャンバスに `.qlr` ファイルをドラッグアンドドロップすることもできます。

12.6 データのドキュメント作成

レイヤでデータを表示したり、シンボルを作成するだけでなく、QGIS ではレイヤに以下の情報を入力することができます：

- **メタデータ** ：データセットの入手方法や内容の理解に役立つ情報、アクセス方法、使用方法などに関する情報です。これらの情報はデータソースのプロパティであり、QGIS のプロジェクトとは切り離して存在可能なものです。
- **レイヤノート** ：現在のプロジェクトにおけるレイヤに関する説明やコメント

12.6.1 メタデータ

レイヤプロパティダイアログの  **メタデータ** タブには、レイヤに関するメタデータを作成・編集するためのオプションがあります。

入力する情報は、以下に関するものです：

- データの **識別** ：データセットの基本属性（親識別子、識別子、タイトル、要約、言語など）
- データが属する **カテゴリ** ： **ISO** カテゴリと併せて、カスタムのカテゴリを追加することもできます
- **キーワード** ：データおよび関連する概念を標準的な基本語彙に従って検索するためのキーワード
- データセットへの **アクセス** （ライセンス、権利、料金および制約）
- データセットの **領域** ：空間的（CRS、地図領域、高度）な範囲と、時間的な範囲の両方
- データセットの所有者の **連絡先**
- 付随的なリソースや関連する情報への **リンク**

- データセットについての **履歴**

入力された情報の概要は **検証** タブに表示され、フォームに関連した潜在的な問題を特定するのに役立ちます。問題を修正してもよいですが、無視することもできます。

メタデータはデフォルトでプロジェクトファイルに保存されます。 **メタデータ** ドロップダウンには `.qmd` ファイルからメタデータを読み込む/保存するオプションと、「デフォルト」の場所からメタデータを読み込む/保存するオプションがあります。

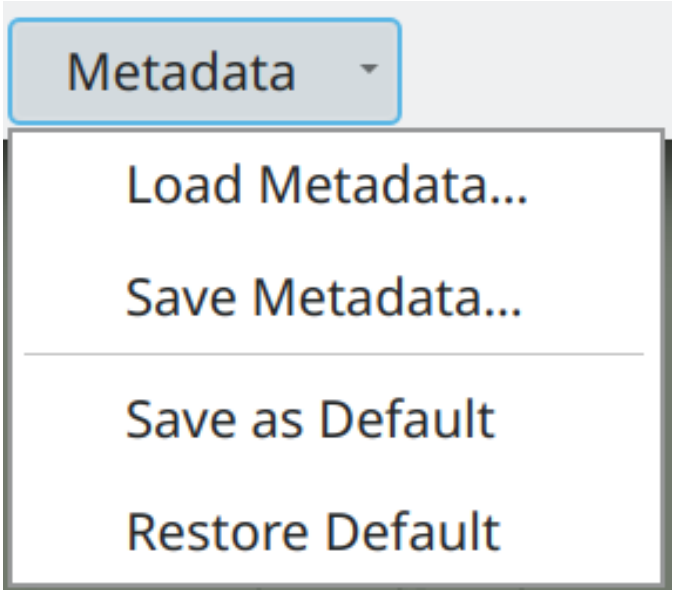


図 12.14: メタデータ読み込み／保存オプション

デフォルトとして保存 と **デフォルトに戻す** で使用される「デフォルト」の場所は、元となっているデータソースとその設定によって変わります：

- PostgreSQL データソースの場合、設定オプション *QGIS* **レイヤメタデータをデータベースに保存/読み込みを許可する** がチェックされていると、メタデータはデータベース内の専用テーブル内に保存されます。
- GeoPackage データソースの場合、**デフォルトとして保存** は常に GeoPackage の内部メタデータテーブルにメタデータを保存します。

  メタデータが PostgreSQL や GeoPackage の内部テーブルに保存されると、ブラウザや :ref:`レイヤ<layer_metadata_search_panel>メタデータ検索パネル` で検索やフィルタリングが可能になります。
- 他のすべてのファイル型のデータソースでは **デフォルトとして保存** はメタデータをファイルと一緒に `.qmd` ファイルに保存します。
- 他のすべての場合 **デフォルトとして保存** はメタデータをローカルの `.sqlite` データベースに保存します。

12.6.2 レイヤノート

レイヤノートは、現在のプロジェクト内でレイヤのドキュメントを作成できます。これは、to do リストや説明、警告など、プロジェクトのユーザーにとって重要なメッセージを保存する場所として使用できます。

レイヤ パネルのレイヤのコンテキストメニューから **レイヤノートを追加**... を選択し、開いたダイアログに必要なテキストを入力します。

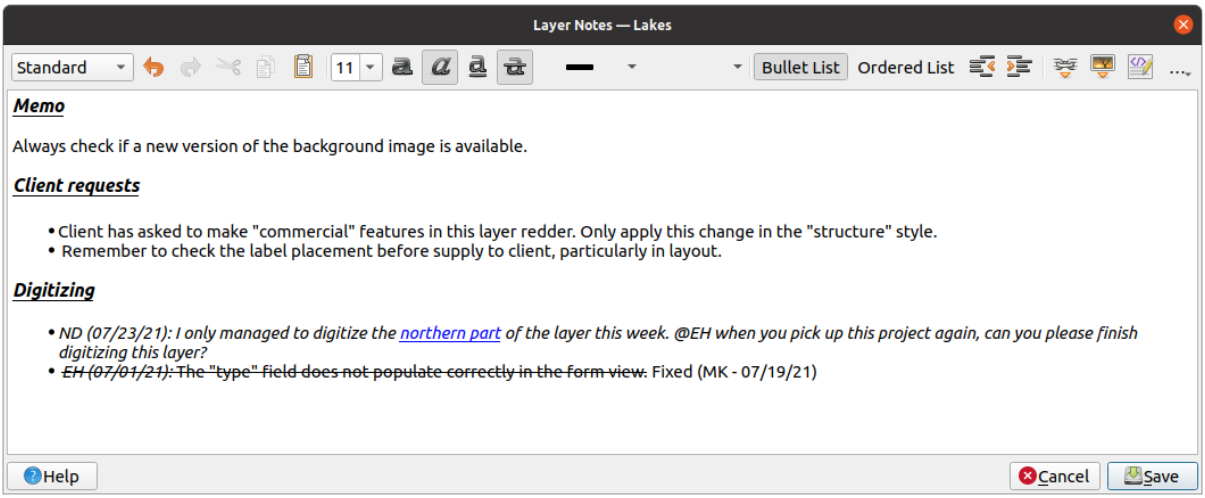


図 12.15: レイヤノートを追加

レイヤノート ダイアログは HTML ベースのマルチラインテキストボックスで、以下のツール一式を備えています：

- テキスト操作：切り取り、コピー、貼り付け、元に戻す、やり直す
- コンテンツの全部または一部に適用できる文字の書式設定：フォントのサイズと色、太字、イタリック、下線、取り消し線、背景色、URL のハイライト
- パラグラフの構造化：箇条書き、順序リスト、字下げ、定義済みの見出し
- ファイルの挿入（ドラッグ＆ドロップでも可）
- HTML コードの編集

ツールバー右端にある ... のドロップダウンメニューでは、以下の操作ができます：

- **すべてのフォーマットを削除**
- **文字フォーマットを削除**
- **内容を消去**

レイヤ パネルにおいて、レイヤノートを持つレイヤには アイコンが付き、このアイコンのマウスを乗せるとノートが表示されます。このアイコンをクリックすると、レイヤノートを編集できます。また、レイヤを右クリックして **レイヤノートを編集**... したり、 **レイヤノートを削除** することもできます。

注釈: レイヤノートは **レイヤスタイル** の一部であり、 `.qml` ファイルや `.qlr` ファイルに保存できます。レイヤスタイルのコピー＆ペーストによって、あるレイヤのノートを別のレイヤに移転させることもでき

ます。

12.7 値を変数に格納する

QGIS では、繰り返し使用する便利な値（例えばプロジェクトのタイトル、ユーザーの氏名など）を変数に保存し、式で使用できます。変数は、アプリケーションのグローバルレベル、プロジェクトレベル、レイヤレベル、プロセシングモデルレベル、レイアウトレベル、レイアウトアイテムのレベルで定義できます。CSS のカスケードルールと同様に、変数は上書きされます。例えば、プロジェクトレベルの変数は、同じ名前で設定されたアプリケーショングローバルレベルの変数を上書きします。変数名の前に @ 文字を付けることで、変数を使用してテキスト文字列やその他のカスタム式を作成できます。例えば、印刷レイアウトのラベル作成で以下の内容とすると：

```
This map was made using QGIS [% @qgis_version %]. The project file for this
map is: [% @project_path %]
```

次のようなラベルをレンダリングします:

```
This map was made using QGIS 3.4.4-Madeira. The project file for this map is:
/gis/qgis-user-conference-2019.qgs
```

プリセットの読み取り専用変数 の他に、上記のどのレベルに対しても独自のカスタム変数を定義することができます。以下の場所で変数を管理できます：

- **グローバル変数 設定　オプション** メニューから
- **プロジェクト変数 プロジェクトのプロパティ** ダイアログ（**プロジェクトのプロパティ** 参照）から
- **ベクタレイヤ変数 レイヤプロパティ** ダイアログ（**ベクタプロパティダイアログ** 参照）から
- **モデル変数 モデルデザイナー** ダイアログから（:ref:`processing.modeler`を参照）；
- **レイアウト変数** 印刷レイアウトの **レイアウト** パネル（**レイアウトパネル** 参照）から
- **レイアウトアイテム変数** 印刷レイアウトの **アイテムプロパティ** パネル（**レイアウトアイテムの共通オプション** 参照）から

編集可能な変数と区別するために、読み取り専用変数の名前と値はイタリック体で表示されます。一方、低いレベルのもので上書きされた高レベル変数には取り消し線が引かれます。

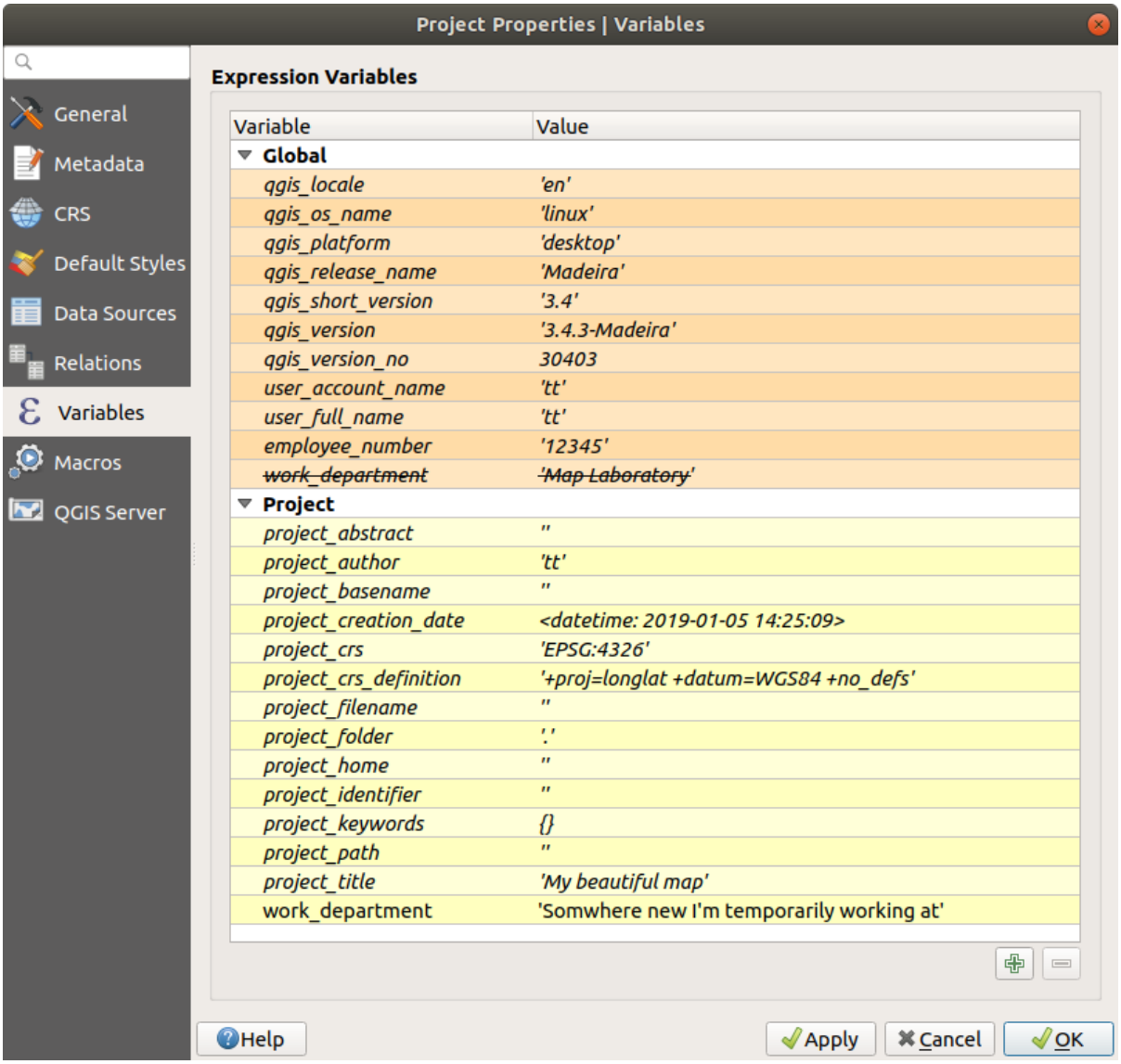


図 12.16: プロジェクトレベルの変数エディタ

注釈: 変数についての詳細や例を知りたい場合には、 Nyall Dawson 氏のブログ投稿 Exploring variables in QGIS 2.12, part 1 、 part 2 、 part 3 を参照してください。

12.8 認証

QGIS には認証資格情報を安全な方法で保存/取得する機能があります。ユーザは認証情報を認証設定に安全に保存することができます。この認証情報はポータブルデータベースに保存され、サーバまたはデータベース接続に適用され、プロジェクトまたは設定ファイル内の ID トークンによって安全に参照されます。詳細については **認証システム** を参照してください。

マスターパスワードは、認証システムとそのポータブルデータベースを初期化する際に設定する必要があります。

12.9 共通のウィジェット

QGIS では、作業頻度が多いオプションがいくつかあります。作業に便利なように、QGIS では以下に紹介する特別なウィジェットが用意されています。

12.9.1 カラーセレクタ

色ダイアログ

色を選ぶために  をクリックすると、 **色の選択** ダイアログが現れます。このダイアログの機能は **設定** **オプション...** **一般情報** の **ネイティブの色選択ダイアログを使用する** パラメータのチェックボックスの状態に依存します。チェックが入っている場合は、色選択には QGIS が起動している OS のネイティブの色選択ダイアログが使用されます。チェックが入っていない場合には、QGIS のカスタムカラーセレクタが使われます。

このカスタムカラーセレクタダイアログは 4 つの異なるタブを持っており、 カラーランプ 、 カラーホイール 、 色見本 または カラーピッカー によって色を選択することができます。前 2 者のタブでは、可能な全ての色組み合わせを閲覧し、選択した色をアイテムに適用することができます。

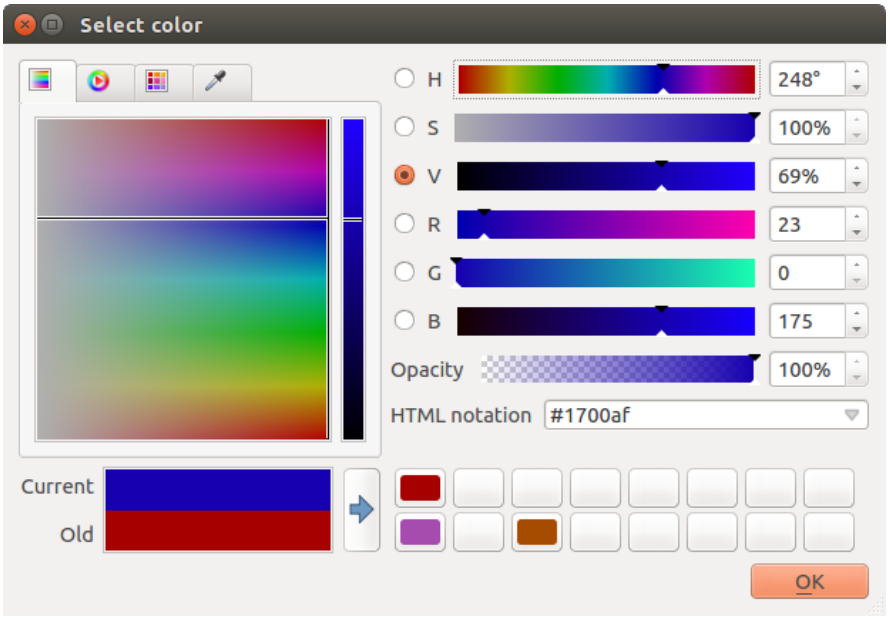


図 12.17: カラーランプタブ

色見本 タブでは、カラーパレットのリストから色を選択することができます（詳細は **色の設定** 参照）。 **最近使った色** パレット以外は、フレームの下部にある カレント色を追加 と 選択した色の削除 ボタンで編集することができます。

パレットのコンボボックスの隣にある ... ボタンは以下のオプションを提供します。

- 色のコピー、貼り付け、インポートまたはエクスポート
- カラーパレットの新規作成、インポートまたは削除

- **色ボタンに表示** の項目で、カスタムパレットをカラーセレクタウィジェットに追加する（ 図 12.19 参照）

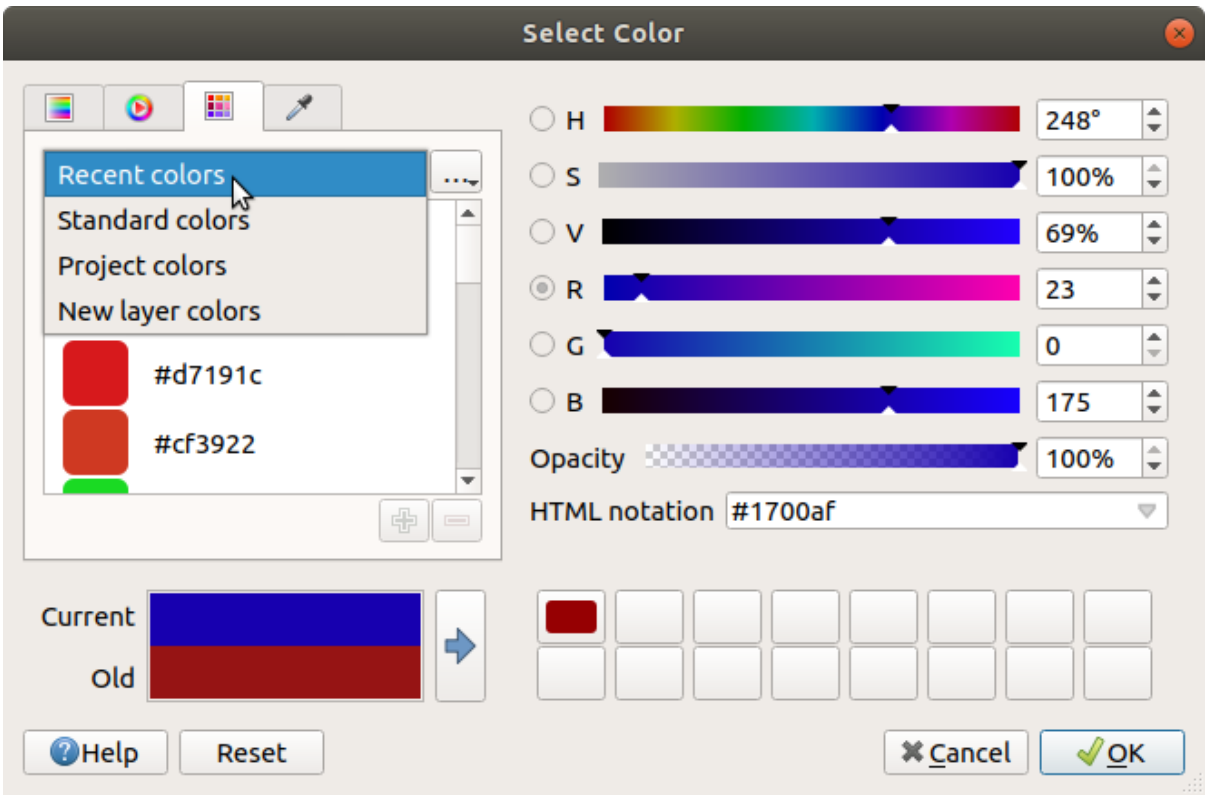


図 12.18: 色見本タブ

別の選択肢として、 カラーピッカー を使用して QGIS の UI や他のアプリケーションの任意の部分に置いたマウスカーソルの下にある色をサンプルすることができます。タブがアクティブなときにスペースバーを押し、目的の色の上にマウスを移動させて、クリックするかスペースバーを再度押します。 **色をサンプルする** ボタンをクリックしても、カラーピッカーを起動することができます。

どの方法を使ったとしても、選択された色はカラースライダーの HSV（Hue, Saturation, Value）と RGB（Red, Green, Blue）の値で表現されます。また、色は *HTML* **表記法** でも確認できます。

色の変更は、カラーホイールやカラーランプ、任意の色パラメータのスライダーをクリックすることで簡単に行えます。色パラメータは横のスピンボックスを使うか、対応するスライダー上でマウスホイールをスクロールすることで調整することができます。また、HTML 表記法で色を入力することもできます。最後に、透明度を設定するための **不透明度** スライダーがあります。

このダイアログには（オブジェクトに適用されている） **前の色** と（選択した） **現在の色** を視覚的に比較する機能もあります。ドラッグドロップするか、 見本に色を追加 ボタンを押すと、これらの色をスロットに保存し、簡単にアクセスできます。

Tip: 色のクイック変更

あるカラーセレクタウィジェットを別のカラーセレクタウィジェットへドラッグ＆ドロップすると、その色を適用することができます。

色選択のドロップダウンショートカット

 色ボタンの右にあるドロップダウンの矢印をクリックすると、色を素早く選択するためのウィジェットが表示されます。このショートカットには、以下の機能があります：

- 色を選ぶためのカラーホイール
- 色の不透明度を変更するためのアルファ値スライダー
- **色ボタンに表示** で設定されたカラーパレット
- 現在の色のコピーと、別のウィジェットへの貼り付け
- コンピュータ画面上の任意の場所から色を取得
- カラーセレクタダイアログから色を選択する
- 色をあるウィジェットから別のウィジェットにドラッグ＆ドロップして、素早く色を変更する

Tip: 色選択ウィジェット上でマウスホイールをスクロールすると、関連する色の不透明度をすばやく変更できます。

注釈: カラーウィジェットがデータによって定義された上書きプロパティによって **プロジェクトの色** に設定されている場合、上記の色を変更するための機能は利用できません。まず初めに **色のリンクの解除** をするか、定義を **クリア** する必要があります。

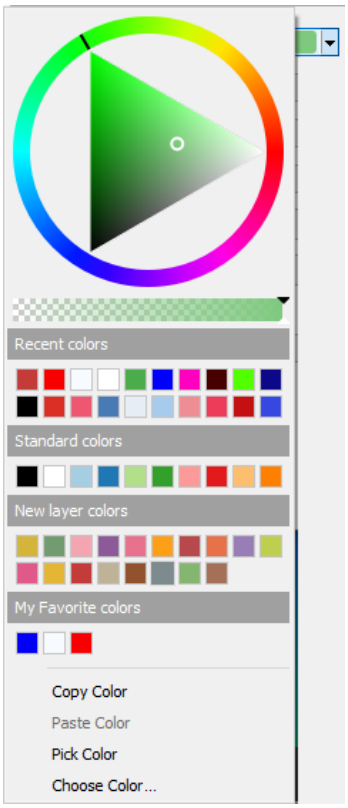


図 12.19: クイックカラーセレクタメニュー

カラーランプのドロップダウンショートカット

カラーランプは、1 つまたは複数の地物に色の組み合わせを適用するための実用的な方法です。その作成方法については、 **カラーランプの設定** のセクションで説明しています。色については、  カラーランプボタンを押すと、対応したカラーランプタイプのダイアログが開き、そのプロパティを変更することができます。

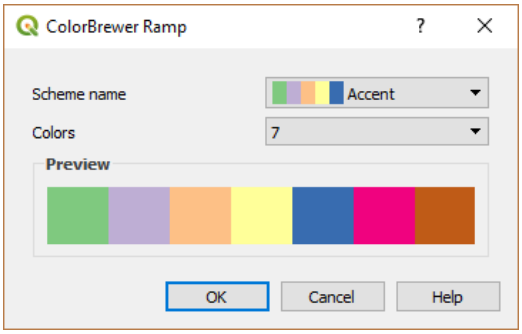


図 12.20: ColorBrewer ランプをカスタマイズする

ボタンの右側にあるドロップダウンメニューを使用すると、カラーランプとオプションの幅広いセットに素早くアクセスできます：

- **カラーランプを反転**

- **現在のランプをクリア** ウィジェットに割り当てられているカラーランプを解除します（特定の箇所で利用可能）
- **ランダムカラーランプ** ：一部の状況でのみ利用可能で（例えばカラーランプをレイヤのシンボロジに使用する場合）、このエントリにチェックを入れるとランダムな色でカラーランプを作成し適用します。また、 **色をシャッフル** エントリを使用でき、現在のカラーランプに満足できない場合には新しいランダムカラーランプを再生成できます。
- **スタイルマネージャ** ダイアログにおいて **お気に入り** に設定されている グラデーション あるいは カタログ: `cpt-city` のカラーランプのプレビュー
- **全カラーランプ** で、対応するカラーランプデータベースへアクセスする
- **カラーランプを新規作成...** で、現在のウィジェットで使用可能な、任意のサポートされている形式のカラーランプを作成する（このカラーランプは、ライブラリに保存しない限りは他の場所で使用できないことに注意してください）
- **カラーランプを編集...** は、カラーランプボタンをクリックした時と同じです。
- **カラーランプを保存...** は、現在のカラーランプをカスタマイズが適用された状態でスタイルライブラリに保存します

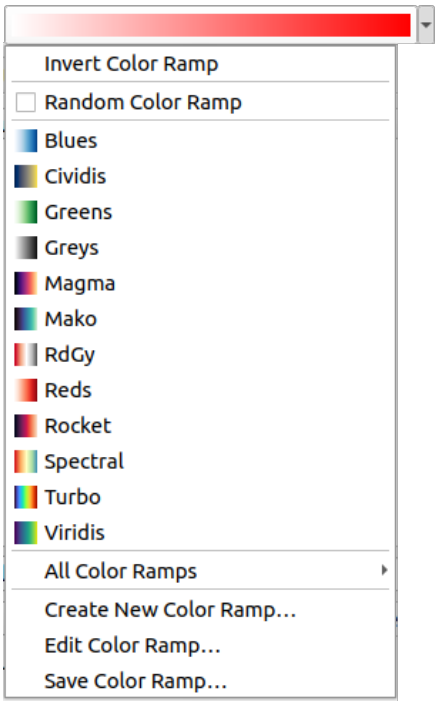


図 12.21: クイックカラーランプ選択ウィジェット

12.9.2 シンボルウィジェット

シンボル セレクタウィジェットは、地物のシンボルプロパティを設定したい場合に便利なショートカットです。ドロップダウン矢印をクリックすると、以下のシンボルオプションが **色ドロップダウンウィジェット** の機能とともに表示されます。

- **シンボルの設定**... ： これはシンボルセレクタウィジェットを起動するのと同じです。 **シンボルのパラメータ** を設定するためのダイアログを開きます。
- **シンボルをコピーする** は、現在のアイテムのシンボルをコピーします
- **シンボルを貼り付ける** は、現在のアイテムにシンボルを貼り付けます。素早くシンボル設定ができます
- **現在のシンボルをクリア** ウィジェットに割り当てられているシンボルを解除します（特定の箇所で利用可能）

Tip: マーカー又はラインシンボルウィジェット上でマウスホイールをスクロールすると、関連するシンボルの大きさをすばやく変更できます。

12.9.3 リモート / 埋め込みファイルセレクタ

ファイルセレクタウィジェットの横に、... ボタンと一緒にドロップダウンの矢印ボタンが表示されていることがあります。これは、以下の箇所で利用できます:

- シンボルやラベルに SVG ファイルを使用する場合
- シンボルやラベル、テクスチャ、地図整飾のカスタマイズにラスタ画像を使用する場合

矢印をクリックすると、以下の操作ができるメニューが表示されます:

- ファイルを選択：ファイルはファイルパスを通じて識別され、QGIS はパスを解決して対応する画像を表示します
- URL から：上と同様ですが、画像はリモートのリソースから取得して読み込みます
- ファイルを埋め込む：ファイルを現在のプロジェクトやスタイルデータベース、印刷レイアウトのテンプレート等に埋め込みます。これにより、そのファイルはアイテムの一部として常にレンダリングできるようになります。これはカスタムシンボルを含んだ自己完結型のプロジェクトを作成するのに便利な方法であり、異なるユーザーや QGIS インストールの間で簡単にカスタムシンボルを共有できます。
- 埋め込みファイルを抽出：ウィジェットを通じて埋め込みファイルを抽出し、ディスク上に保存します

12.9.4 表示縮尺セレクタ

表示縮尺セレクタは、要素をマップキャンバスに表示する縮尺を制御するオプションを提供します。指定した縮尺の範囲の外では、要素は表示されません。**レンダリング** プロパティタブから、レイヤ、ラベル、ダイアグラムなどに適用できます。

1. **縮尺に応じた表示設定** ボックスをチェックします
2. **最小縮尺（含まない）** ボックスに、値を入力するか **定義済み縮尺** から選択して、要求する最も縮小したときの縮尺を入れます
3. そして／または :guilabel:最大縮尺（含む）' ボックスに、要求する最も拡大したときの縮尺を入れます

   縮尺ボックスの隣にある 現在キャンバスが表示している縮尺を設定 ボタンは、現在のマップキャンバスの縮尺を　表示範囲の境界として設定します。ボタンの横の矢印を押すと、レイアウトのマップから縮尺にアクセスし、ボックスへの入力に再利用できます。

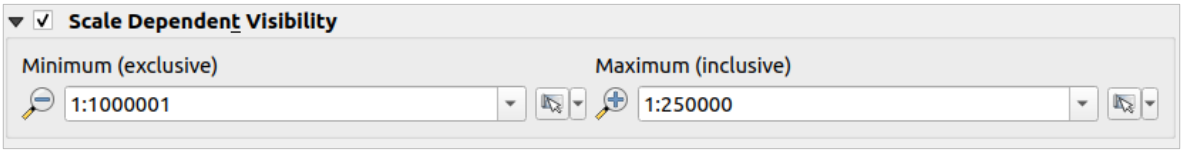


図 12.22: 表示縮尺セレクタウィジェット

12.9.5 空間範囲セレクタ

範囲 セレクタウィジェットは、空間的な範囲を選択して、レイヤに割り当てたり実行するアクションの範囲を制限したい場合に便利なショートカットです。使用される状況にもよりますが、以下の中から範囲を選択できます:

- **現在のレイヤの領域**: 例 レイヤをエクスポートする場合
- **レイヤから計算**　: 現在のプロジェクトに読み込まれたレイヤの範囲を使います
- 現在の **キャンバスの領域** を使います
- **キャンバスに描画**: 矩形を描画してその座標を使います
- **ブックマークから計算**: 保存されている **ブックマーク** の範囲を使います
- **レイアウトマップから計算**: **レイアウトマップ** の範囲を使います
- `xmin, xmax, ymin, ymax` の形式で座標を入力または編集する

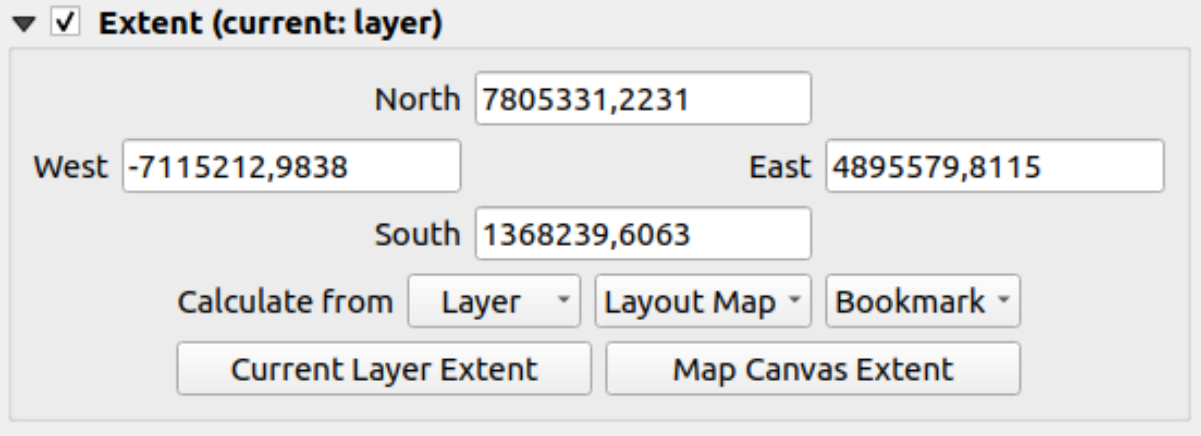


図 12.23: 範囲セレクタウィジェット

12.9.6 フォントセレクタ

フォント セレクタウィジェットは、テキスト情報（地物のラベル、地図整飾のタイトルラベル、地図の凡例テキスト等...）に対してフォントプロパティを設定したい場合に便利なショートカットです。ドロップダウン矢印をクリックすると、以下のオプションの一部または全てが表示されます。

図 12.24: フォントセレクタのドロップダウンメニュー

- **現在のテキストフォーマットをクリア** ウィジェットに割り当てられているテキストフォーマットを解除します（特定の箇所で利用可能）
- **フォントサイズ** を関連づけられた単位で設定します

- **最近使ったフォント**　メニューは、最近使ったフォントと共に、アクティブなフォントが（最上部に）チェックされた状態で表示されます
- **フォーマットを設定**... はフォントセレクタウィジェットのボタンを押した場合と同様です。テキストフォーマットのパラメータを設定するダイアログが開きます。テキスト設定の状況にもよりますが、このダイアログは OS のデフォルトの **テキストフォーマット** ダイアログ、もしくは、**ラベルテキストの書式設定** のセクションで説明する、高度なフォーマットのオプション（不透明度、テキストの向き、バッファ、背景、影など...）を備えた QGIS のカスタムダイアログです。
- **コピー形式** ：テキストのフォーマットをコピーします
- **貼り付けフォーマット** ：コピーしたフォーマットをテキストに貼りつけ、フォーマットを素早く設定できます
- **色ウィジェット** ：簡単に色設定ができます

Tip: フォント選択ウィジェット上でマウスホイールをスクロールすると、関連するテキストのフォントサイズをすばやく変更できます。

12.9.7 単位セレクタ

QGIS のアイテム（ラベル、シンボル、レイアウト要素 ...）のサイズプロパティは、必ずしもプロジェクト単位または特定のレイヤの単位に拘束されるわけではありません。多くのプロパティでは、**単位** セレクタのドロップダウンメニューを使用して、必要なレンダリングに応じて（画面解像度、用紙サイズ、または地形を基準として）プロパティの値を調整することができます。利用可能な単位は以下のとおりです：

- **ミリメートル（***Millimeters***）**
- **ポイント（***Points***）**
- **ピクセル（***Pixels***）**
- **インチ（***Inches***）**
- **パーセント**: あるプロパティを別のプロパティのパーセントとして設定することができます。例えば、バッファ／影の大きさを一定にする代わりに、テキストの大きさの変更に応じて構成要素（バッファの大きさ、影の半径...）をうまく拡大縮小するテキストフォーマットを作成するのに便利です。そのため、テキストの大きさが変わっても、それらの大きさを調節する必要がありません。
- **縮尺済みメートル（***Meters at Scale***）** ：これにより、基礎となる地図単位（例：インチ、フィート、度 ...）に関係なく、常にサイズをメートル単位で設定できます。メートル単位のサイズは、現在のプロジェクト楕円体の設定と、現在のマップ範囲の中心位置でのメートル単位の距離の投影に基づいて計算されます。投影座標系のマップの場合、この設定は投影された単位長さを使用して計算されます。地理的座標系（緯度/経度）に基づくマップの場合には、サイズはマップの縦方向スケールの楕円体計算を使用して計算されたメートル単位で近似されます。
- **地図単位** ： サイズはマップビューのスケールに応じて拡大縮小されます。これは大きすぎる、あるいは小さすぎる値となることがあるため、エントリの隣にある 🔧 ボタンを使用して、サイズを以下に基づいた値の範囲に制限します：

- **最小縮尺** と **最大縮尺** ：これらの縮尺限界値に到達するまでは、値はマップビューの縮尺に基づいてスケーリングされます。縮尺の範囲外の場合は、値は最も近い縮尺限界値に基づく値のままです。
- かつ/または **最小サイズ** と **最大サイズ** を mm 単位で指定したもの ：値はこれらのサイズ限界値に到達するまで、マップビューの縮尺に基づいてスケーリングされます。限界値に到達した場合は、このサイズ限界値で一定となります。

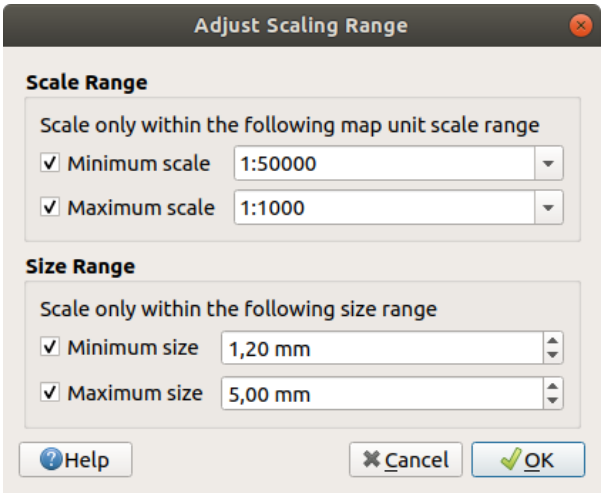


図 12.25: 縮尺範囲の調整ダイアログ

12.9.8 数値フォーマット

数値フォーマットでは、様々な整形方法（例えば指数表記、通貨、パーセント等）を使用して数値の表示を整形できます。この使用例には、レイアウトのスケールバーや固定サイズのテーブルのテキストなどがあります。

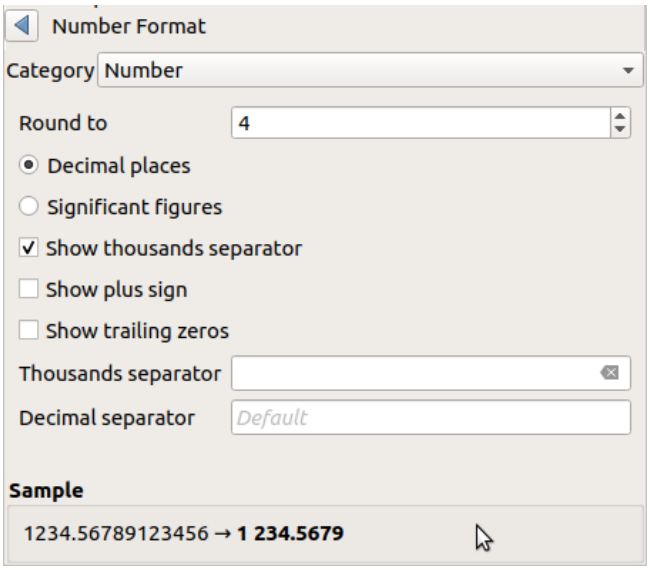


図 12.26: 数値フォーマット

様々なカテゴリのフォーマットをサポートしています。多くの場合、下記の数値オプションの一部もしくはすべてを設定できます：

- ☑ **数字の３桁区切り記号を表示**
- ☐ **正の符号を表示**
- ☐ **ゼロ埋め表示**

ただし、カスタム設定もあります。提供されているカテゴリは以下のとおりです。

- **一般情報** 、デフォルトのカテゴリ：設定がなく、親ウィジェットの設定もしくはグローバル設定と同様に値を表示します。
- **数値**
  - 値はユーザーが定義した **小数点以下桁数** もしくは **有効数字** で *:guilabel:* **数値を丸める** ことができます
  - **数字の３桁区切り記号** や **小数点記号** をカスタマイズできます
- 方向/方位のテキスト表現を設定する **方位** には、以下の設定があります。
  - **形式** ：利用可能な値の範囲は `0` から `180°`, （`E/W` 付き）、`-180` から `+180°` 、`0` から `360°` です
  - **小数点以下桁数** の数字
- **通貨** は、通貨の値のテキスト表現のための設定です。
  - **接頭辞**
  - **単位文字**
  - **小数点以下桁数** の数字
- **分数** は、小数値の分数表現のための設定です（例： *0.5* の代わりに *1/2* ）
  - 表示に ☐ *Unicode* **分数表現を使用** する。 例えば、1/2 の代わりに $^{1}/_{2}$ を使用します
  - ☐ **専用** *Unicode* **文字を使用**
  - **数字の** *3* **桁区切り記号** をカスタマイズする
- **パーセント** - 値に `%` を付け足します。以下の設定があります
  - **小数点以下桁数** の数字
  - **縮尺** ：実際の値がすでにパーセント表現となっている（そのままの値）か、小数値（パーセントに変換する）かを指示します
- **指数表記** は、 `2.56e+03` の形式での表記です。 **小数点以下桁数** の数字を指定できます。

サンプル セクションの下には、設定のライブプレビューが表示されます。

12.9.9 混合モード

QGIS は混合モードツールを通じて、これまでは画像編集プログラムでしか知られていないような、特別なレンダリング効果のためのさまざまなオプションを提供しています。混合モードは、レイヤや地物に適用したり、印刷レイアウトアイテムにも適用することができます：

- **通常**（*Normal*）：これは標準の混合モードで、上にあるピクセルのアルファチャンネルを使用して、その下のピクセルと混合します。色は混合されません。
- **比較・明**（*Lighten*）：これは、前景と背景のピクセルから各成分の最大値を選択します。結果はギザギザで粗くなる傾向があることに注意してください。
- **網掛け処理**（*Screen*）：アイテムの明るいピクセルは下のレイヤ上に塗られますが、暗いピクセルは塗られません。このモードは、あるアイテムのテクスチャを別のアイテムに混ぜるのに便利です（例えば、陰影図を使って他のレイヤのテクスチャを作成するなど）。
- **覆い焼き**（*Dodge*）：上にあるピクセルの明るさに基づき下のピクセルを明るくし、彩度を上げます。上にあるピクセルが明るいほど、下のピクセルの彩度と明るさが上がります。これは、上にあるピクセルが明るすぎない場合に最適です。そうでない場合は効果が極端になります。
- **覆い焼き・加算**（*Addition*）：あるアイテムのピクセル値を他方のピクセル値に加えます。値が上限値よりも大きくなる場合、（RGB モードの場合は）白色で表示されます。このモードは地物をハイライトするのに適しています。
- **比較・暗**（*Darken*）：上と下のピクセルの各成分で最小値を保持します。「比較・明」と同様に、結果はギザギザで粗くなる傾向があります。
- **乗算**（*Multiply*）：上のアイテムのピクセル値を下のアイテムのピクセル値と掛け合わせます。結果は暗くなります。
- **焼きこみ**（*Burn*）：上にあるアイテムの色が濃いと、下にあるアイテムの色が濃くなります。「焼きこみ」は下にあるレイヤの色を調整したり、色付けしたりするのに使えます。
- **オーバーレイ合成**（*Overlay*）：「乗算」と「網掛け処理」混合モードの組み合わせです。明るい部分はより明るく、暗い部分はより暗くなります。
- **ソフトライト**（*Soft light*）：「オーバーレイ合成」とよく似ていますが、「乗算」/「網掛け処理」の代わりに「焼き込み」/「覆い焼き」を使います。これは、画像に柔らかい光が当たっているのを模擬したものです。
- **ハードライト**（*Hard light*）：「ハードライト」も「オーバーレイ合成」モードと非常によく似ています。画像に非常に強い光が当たっているのを模擬したものです。
- **差の絶対値**（*Difference*）： 結果が常に正の値となるように、下のピクセル値から上のピクセル値を引く、もしくはその逆を行います。黒色との混合は、全ての色に対する違いがゼロのため、何の変化もありません。
- **減算**（*Subtract*）：あるアイテムのピクセル値から他のアイテムのピクセル値を引きます。負の値となる場合には、黒く表示されます。

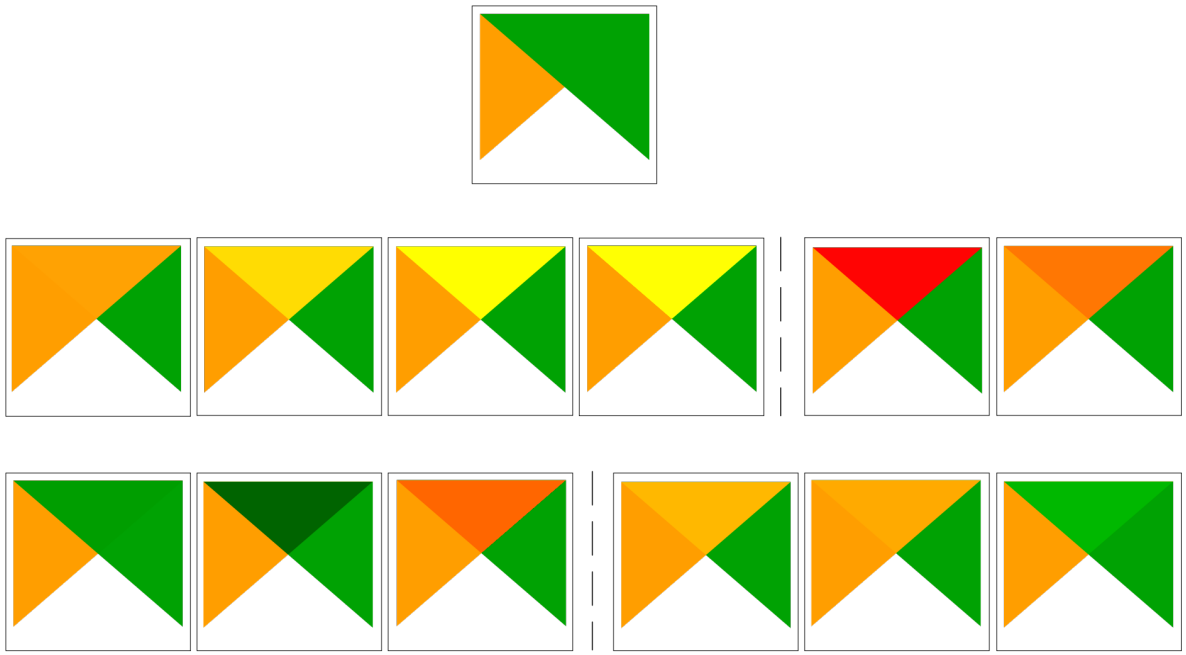


図 12.27: オレンジの地物の上に緑の地物があるときに適用した混合モードの例
上から下、左から右：通常（Normal）-- 比較・明（Lighten）、網掛け処理（Screen）、覆い焼き（Dodge）、加算（Addition） -- 差の絶対値（Difference）、減算（Subtract）-- 比較・暗（Darken）、乗算（Multiply）、焼きこみ（Burn）-- オーバーレイ合成（Overlay）、ソフトライト（Soft light）、ハードライト（Hard light）

レイヤが **グループとしてレイヤを描画** グループの一部である場合、レンダリングに追加の混合モードが利用できます。これらは、あるレイヤの内容のレンダリングを、2 つ目の「マスク」レイヤの内容でクリップする方法を提供します。

- **以下でマスク**： 上のピクセルが出力され、その不透明度は下のピクセルにより減じられます。
- **以下をマスク**： 下のピクセルが出力され、その不透明度は上のピクセルにより減じられます。
- **以下で反転マスク**: 上のピクセルが出力され、不透明度は下のピクセルの反転によって減じられます。
- **以下を反転マスク**: 下のピクセルが出力され、その不透明度は上のピクセルの反転によって減じられます。
- **以下の内部部分を描画**: 上のピクセルは下のピクセルの上に混合され、上のピクセルの不透明度は下のピクセルの不透明度によって減じられます。
- **以下の内側を描画**: 下のピクセルは上のピクセルの上に混合され、下のピクセルの不透明度は上のピクセルの不透明度によって減じられます。

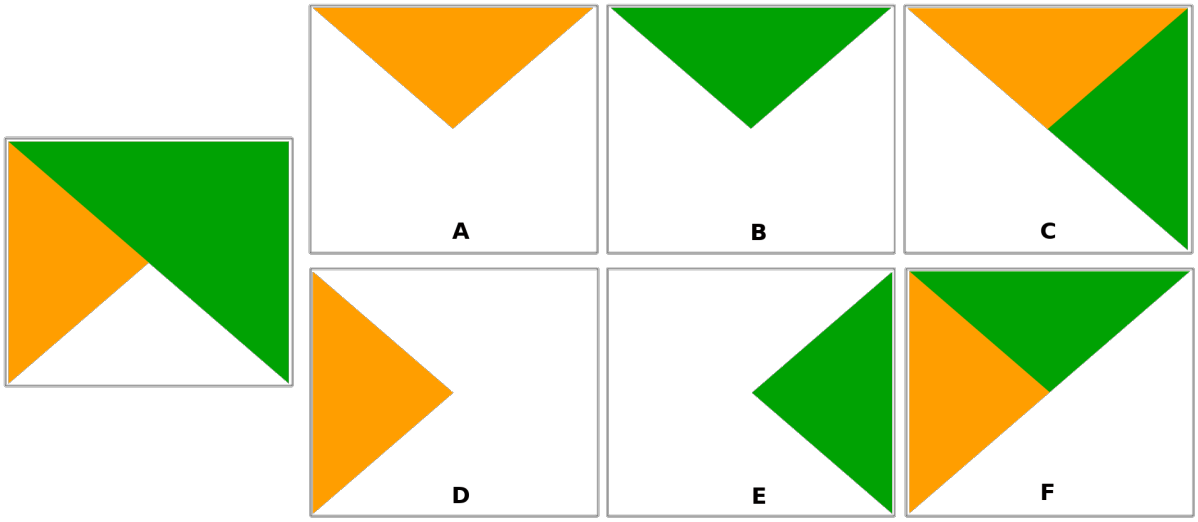


図 12.28: グループ内の上の緑のレイヤに適用された混合切り抜きモードの例
A: 以下をマスク **B**: 以下でマスク **C**: 以下の内側を描画 **D**: 以下を反転マスク **E**: 以下で反転マスク **F**: 以下の内部部分を描画

12.9.10 データによって定義された上書きの設定

ベクタレイヤのプロパティダイアログの多くのオプションや印刷レイアウトの設定の横には、データによって定義された上書き アイコンがあります。このツールでは、レイヤ属性やアイテム設定に基づいた **式** 、予め用意された関数やカスタム関数、 **変数** を使用して、パラメータに動的な値を設定することができます。これを有効にすると、通常の値（チェックボックス、テキストボックス、スライダー...）に関係なく、このウィジェットによって返される値がパラメータに適用されます。

データによって定義された上書きウィジェット

データによって定義された上書き アイコンをクリックすると、以下のエントリが表示されます：

- **説明**... は、オプションが有効化されているか、予想される入力値、有効な入力型、そして現在の定義を表示します。ウィジェットの上にマウスカーソルを乗せることでも、この情報を表示します。
- **データをプロジェクトに格納する** ： **補助テーブルプロパティ** 機能を使用してプロパティをプロジェクトに保存することを許可するためのボタンです。
- **フィールドの型**: レイヤのフィールドのうち、有効な入力型に一致するフィールドから選択するためのエントリです。
- **色**: ウィジェットが色プロパティと関連しているときには、このメニューから現在の **プロジェクトの色** スキームとして定義されている色にアクセスできます。
- **変数** ：利用可能なユーザー定義の **変数** へアクセスするためのメニューです。

- **編集**... ボタンは、 **式文字列ビルダ** ダイアログを使用して、適用する式を作成もしくは編集するためのボタンです。正しく式を入力するのを助けるために、期待される出力フォーマットのヒントがダイアログ内に表示されています。
- **貼り付け** と **コピー** ボタン
- **クリア** ボタンで設定を削除します。
- 数値と色のプロパティには、地物データがプロパティにどのように適用されるのかを再スケーリングするための **アシスタント**... があります（詳細については **下記** を参照ください）。

Tip: データによる上書きを右クリックで有効化（無効化）

データによって定義された上書きのオプションが正しく設定されている場合には、アイコンは黄色で  や  になります。もし上書きオプションが壊れている場合には、アイコンは赤色で  や  になります。

ウィジェットをマウスの右ボタンでクリックすることで、設定された  データによって定義された上書き ボタンを簡単に有効化・無効化できます。

データによる定義のアシスタントインターフェースを使用する

 データによって定義された上書き ボタンが大きさや回転、不透明度、色プロパティと関連づけられている場合には、 **アシスタント**... オプションがあり、各地物に対してデータがパラメータに適用される方法を変更する助けとなります。アシスタントには以下の機能があります:

- **入力** データの定義。すなわち：
  - **ソース** ：フィールドまたは  **式** を使用して表現する属性値
  - 表す値の範囲：値を手入力することもできますが、  レイヤから値の範囲を取得する ボタンを使用して、データに適用された **ソース** 式によって返される最小値と最大値でこのフィールドを自動的に入力することもできます。
-  **変換曲線を適用する** ：デフォルトでは、出力値（設定は下記を参照）は線形スケールに従って入力地物に適用されます。 これを変換曲線ロジックを利用して上書きすることができます。変換オプションを有効にして、グラフにブレークポイント（複数可）を追加しポイントをドラッグして、適用するカスタム分布が作成できます。
- **出力** 値の定義：このオプションは定義するパラメータによって異なります。共通して設定できるのは以下の項目です：
  - 色の設定については、値に適用する **カラーランプ** と、値が NULL のときに使用する単一色
  - それ以外の場合は、選択したプロパティに適用する最小値・最大値と、ソース地物の値が無視される場合や NULL のときに出力する大きさ/角度/不透明度の値
  - 大きさプロパティについては、表現の **スケール方法** の設定。 **フラナリー** 、 **指数関数的** 、 **サーフェスグリッド（Surface）** 、 **半径** 、 **線形** の中から選択します

- データのスケーリングに使用する **指数** 。 **スケール方法** を「指数関数的」とする場合や、不透明度の調整において使用します

対応しているプロパティの場合には、ダイアログの右側にライブ更新プレビューが表示され、値のスケーリングを制御するのに役立ちます。

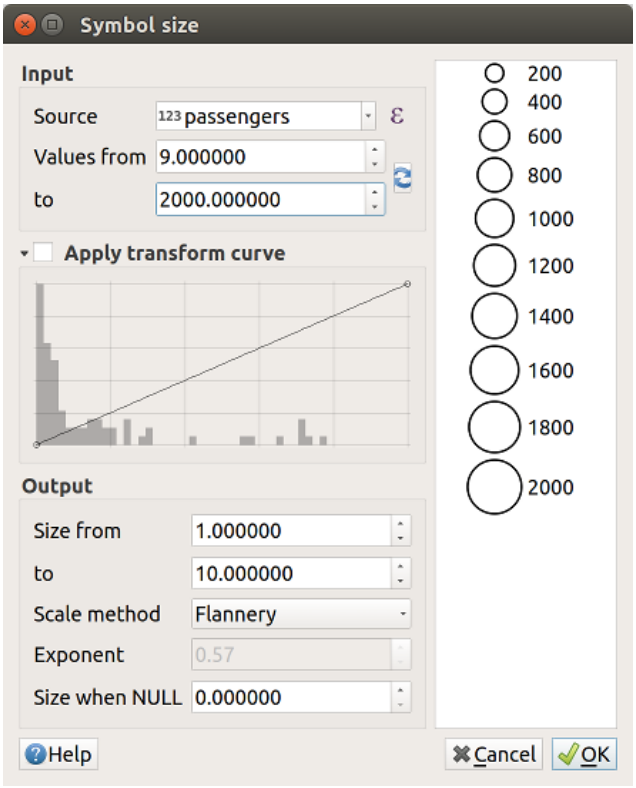


図 12.29: passengers レイヤのフィールド値に基づいた地物サイズのスケーリング

上図のシンボルの大きさアシスタントによって設定された値は、大きさの「データによって定義された上書き」を以下のように設定します:

```
coalesce(scale_exp("passengers", 9, 2000, 1, 10, 0.57), 0)
```

第13章　式でレベルアップ

13.1 式

レイヤのデータと組み込みの関数やユーザー定義の関数に基づいた **式** は、属性値やジオメトリ、変数を操作する強力な方法です。ジオメトリのスタイル、ラベルの内容や位置、ダイアグラムの値、レイアウトアイテムの高さ、地物の一部の選択、仮想フィールドの作成などを動的に変更することができます。

注釈: 式の作成に使えるデフォルトの関数や変数のリストは、詳細な情報や例付きで **関数のリスト** にあります。

13.1.1 式文字列ビルダー

式を組み立てるための主要なダイアログである **式文字列ビルダー** は、QGIS の多くの部分から使用することができますが、特に以下のような時に使うことができます。

-  ボタンをクリックしたとき
-  式による地物選択... ツールで **地物を選択** するとき
- 例えば  フィールド計算機 ツールで **属性を編集する** とき
-  データによって定義された上書き ツールを使用して、シンボロジやラベル、レイアウトアイテムのパラメータを操作するとき（**データによって定義された上書きの設定** 参照）
- **ジオメトリジェネレータ** のシンボルレイヤを作成するとき
- 何らかの **ジオプロセシング** を行うとき

式ビルダーダイアログからは、以下のタブにアクセス可能です。

- **式タブ** は、事前に定義された関数のリストを利用して、使用する式の作成や確認ができます。
- **関数エディタタブ** は、カスタムの関数を作成することで、関数リストの拡張ができます。

インターフェース

式 タブには関数やレイヤのフィールド、値を使用して式を作成するためのメインのインターフェースがあります。これには、以下のウィジェットが含まれます：

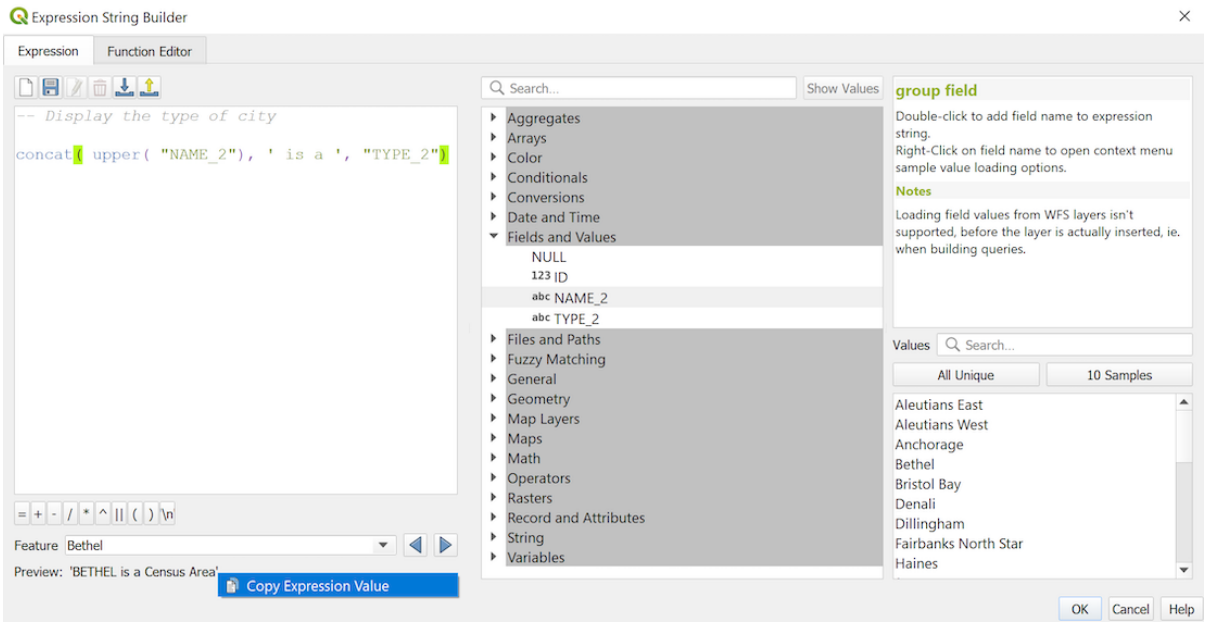


図 13.1: 式タブ

- 式を入力したり張り付けたりするための式エディタ領域。以下のような自動補完を利用でき、式の入力を短縮できます：
  - 入力したテキストに応じた変数名、関数名、フィールド名が下に表示されます。 Up 矢印キーと Down 矢印キーを使って項目間を移動することができます。式に挿入するには Tab キーを押すか、挿入したい項目を単にクリックします。
  - 関数を入力している間、関数のパラメータが表示されます。

  また、QGIS は式が正しいかをチェックし、エラーを以下の方法で強調表示します：
  - **アンダーライン** ：未定義の関数や、間違ったあるいは無効な引数
  - **マーカー** ：単一の場所でのその他エラー（対応していない括弧、不要な文字など）

Tip: コメントで式をドキュメンテーションする

複雑な式を使用する場合には、思い出しやすいように複数行コメントやインラインコメントとしてテキストを追加するのがよいでしょう。

```
/*
Labels each region with its highest (in altitude) airport(s)
and altitude, eg 'AMBLER : 264m' for the 'Northwest Artic' region
*/
with_variable(
  'airport_alti', -- stores the highest altitude of the region
```

(次のページに続く)

(前のページからの続き)

```
  aggregate(
    'airports',
    'max',
    "ELEV", -- the field containing the altitude
    -- and limit the airports to the region they are within
    filter := within( $geometry, geometry( @parent ) )
  ),
    aggregate( -- finds airports at the same altitude in the region
      'airports',
      'concatenate',
      "NAME",
      filter := within( $geometry, geometry( @parent ) )
        and "ELEV" = @airport_alti
    )
    || ' : ' || @airport_alti || 'm'
    -- using || allows regions without airports to be skipped
)
```

- 式エディタの上部には、以下のようなツールセットがあります：
  -  式エディタをクリア
  - **ユーザー式** を作成、管理する
- 式エディタの下部には、以下のものがあります：
  - 式を構築するのに役立つ基本的な演算子のセット
  - データで地物のプロパティを決める際に、出力の期待されるフォーマットの表示
  - デフォルトでレイヤの最初の地物で評価された、式のライブ **プレビュー** (最大 60 文字)。60 文字を超える出力プレビューテキストを表示するには、テキストにカーソルを合わせ、出力プレビュー全体を含むツールチップのポップアップを表示します。出力プレビューテキストをクリップボードにコピーするには、出力プレビューテキストを右クリックして、 **式の値をコピー** を選択します。

    レイヤの他の地物は、 **地物** コンボボックスを使って閲覧・評価することができます（値はレイヤの **表示名** プロパティから取得されます)。

    エラーとなった場合には、エラーが表示され、ハイパーリンクからエラーの詳細情報にアクセスできます。
- 関数セレクタには、関数や変数、フィールドなどのグループ化されたリストが表示されます。検索ボックスを利用して、リストをフィルタし特定の関数やフィールドを素早く見つけることができます。アイテムをダブルクリックすると、そのアイテムが式エディタに追加されます。
- ヘルプパネルには、関数セレクタで選択したアイテムに対するヘルプが表示されます。

Tip: 式内で関数名にカーソルを乗せて `Ctrl+Click` すると、ダイアログでその関数のヘルプが表示

されます。

関数セレクタでフィールドを選択している場合には、フィールドの値ウィジェットが表示され、以下のように地物の属性値の取得に役立ちます。

- 特定のフィールド値を検索する
- **全ユニーク** または *10* **個のサンプル** 値のリストを表示する。これはフィールドの右クリックからも利用できます。

  フィールドが他のレイヤや値のセットにマッピングされている場合、すなわち、 **フィールド編集ウィジェット** が **リレーションの参照** 、 **値のリレーション** または **バリューマップ** タイプの場合には、（参照されたレイヤ、テーブル、リストから）マッピングされたフィールドの値をすべてリストすることができます。さらに、このリストをフィルタリングして、現在のフィールドで **使用中の値のみ表示** することもできます。

ウィジェット内でフィールド値をダブルクリックすると、式エディタに値が追加されます。

Tip: 関数のヘルプやフィールドの値が表示される右側パネルは、ダイアログ内でたたむ（非表示にする）ことができます。 **値を表示** や **ヘルプを表示** ボタンを押すと、非表示状態から表示状態に戻すことができます。

式の作成

QGIS の式は、地物の選択や値の設定に使われます。QGIS で式を作成するには、以下のいくつかのルールに従います。

1. **ダイアログでコンテキストを定義する**: SQL に慣れているならば、 *select features from layer where condition* や、 *update layer set field = new_value where condition* といった形式のクエリを知っているでしょう。QGIS 式もこれらの情報すべてを必要としますが、式ビルダーダイアログを開くためのツールが、必要な情報の一部を提供します。例えば、あるレイヤ（ `buildings` ）が、あるフィールド（ `height` ）を持っているとします:

   - 式による地物選択 ツールを押すことは、 "select features from buildings" を意味します。式テキストウィジェットに入力する必要がある情報は **condition** のみです。つまり、 `"height" > 20` と入力すれば、高さが 20 よりも大きい建物を選択します。
   - この地物選択を行った状態で、 フィールド計算機 ボタンを押して **既存のフィールドを更新する** で "height" を選択することは、コマンド "update buildings set height = ??? where height > 20" を入力したのと同じです。この場合に、残りで入力する必要のあるものは、 **新しい値** だけです。つまり、式エディタのテキストボックスに単に `50` と入力すれば、上で選択した buildings レイヤの height フィールドがこの値に設定されます。

2. **引用符に注意** ：シングルクォートはリテラルを返します。このため、シングルクォートに挟まれたテキスト（ `'145'` ）は文字列として解釈されます。ダブルクォートはそのテキストの値を返すので、フィールドにはダブルクォートを使用してください（ `"myfield"` ）。フィールドは引用符なしでも使用できます（ `myfield` ）。数値には引用符は不要です（ `3.16` ）。

注釈: 関数は通常、フィールド名を表す文字列を引数にとります。次のようにしてください:

```
attribute( @atlas_feature, 'height' ) -- returns the value stored in the "height
↪" attribute of the current atlas feature
```

以下のようにしてはいけません:

```
attribute( @atlas_feature, "height" ) -- fetches the value of the attribute␣
↪named "height" (e.g. 100), and use that value as a field
                                      -- from which to return the atlas feature␣
↪value. Probably wrong as a field named "100" may not exist.
```

Tip: 名前付きパラメータを使用して式を読みやすくする

いくつかの関数は多数のパラメータを設定する必要があります。式エンジンは名前付きパラメータの使用をサポートしています。つまり、暗号めいた式 `clamp( 1, 2, 9)` と書く代わりに、`clamp( min:=1, value:=2, max:=9)` を使うことができます。引数を入れ替え、例えば `clamp( value:=2, max:=9, min:=1)` とすることもできます。名前付きパラメータを使うことで、式の関数の引数が何を指しているのかを明確にすることができ、後で式を理解しようとするときに役立ちます。

式の使用例

- フィールド計算機を使用して、既存の "total_pop" と "area_km2" フィールドを用いた "pop_density" フィールドの計算:

```
"total_pop" / "area_km2"
```

- 面積に基づいた地物のラベル付けやカテゴリ分け:

```
CASE WHEN $area > 10 000 THEN 'Larger' ELSE 'Smaller' END
```

- "pop_density" の値に応じた ”density_level” フィールドのカテゴリ更新:

```
CASE WHEN "pop_density" < 50 THEN 'Low population density'
     WHEN "pop_density" >= 50 and "pop_density" < 150 THEN 'Medium population␣
↪density'
     WHEN "pop_density" >= 150 THEN 'High population density'
END
```

- すべての地物に対して、住宅の平均価格が平方メートル当たり 10000€ より小さいか大きいかに応じたカテゴリ値スタイルを適用:

```
"price_m2" > 10000
```

- 「式による地物選択...」ツールを使用して、“ High population density ” に相当し、加えて平方メートルあたりの住宅の平均価格が 10000€ よりも高いエリアの地物をすべて選択:

```
"density_level" = 'High population density' and "price_m2" > 10000
```

この式は、どの地物をラベル付けするか、あるいはどの地物を地図に表示するかを定義するためにも使用できます。

- ジオメトリジェネレータを使用した、レイヤの様々なシンボル（タイプ）の作成:

```
point_on_surface( $geometry )
```

- ポイント地物に対して、ジオメトリの周囲に閉じた輪を生成（ make_line 関数を利用）:

```
make_line(
  -- using an array of points placed around the original
  array_foreach(
    -- list of angles for placing the projected points (every 90°)
    array:=generate_series( 0, 360, 90 ),
    -- translate the point 20 units in the given direction (angle)
    expression:=project( $geometry, distance:=20, azimuth:=radians( @element ) )
  )
)
```

- 印刷レイアウトのラベルにおいて、レイアウトの "Map 1" アイテム内にある "airports" 地物の名前を表示:

```
with_variable( 'extent',
               map_get( item_variables( 'Map 1' ), 'map_extent' ),
               aggregate( 'airports', 'concatenate', "NAME",
                          intersects( $geometry, @extent ), ' ,'
                        )
             )
```

式の保存

式エディタフレームの上部にある 現在の式をユーザー保存式に追加 ボタンを使用すると、簡単にアクセスしたい重要な式を保存することができます。この式は中央のパネルの **ユーザー式** グループから利用することができます。ユーザー保存式は **ユーザープロファイル** （ <userprofile>/QGIS/QGIS3.ini ファイル）内に保存され、あらゆるプロジェクトで現在のユーザープロファイルを使用すれば、すべての式ダイアログで利用可能です。

式エディタフレームの上部にあるツール群を使用して、ユーザー式の管理ができます：

- 現在の式をユーザー保存式に追加 ：式をユーザープロファイル内に保存します。ラベルとヘルプ文を追加して、識別しやすくできます。
- ユーザー保存式から選択した式を編集 は、式やヘルプ文、ラベルを編集できます。
- 選択した式をユーザー保存式から削除
- ユーザー式をインポート は、 `.json` ファイルからユーザー式をアクティブなユーザープロファイルのフォルダ内にインポートします。
- ユーザー式をエクスポート は、ユーザー式を `.json` ファイルとしてエクスポートします。ユーザープロファイル `QGIS3.ini` ファイル内のユーザー式をすべて共有することができます。

13.1.2 関数エディタ

関数エディタ タブでは、Python 言語を使って独自の関数を書くことができます。これにより、定義済みの関数ではカバーできない特定のニーズに対処するための、手軽で簡単な手段を手に入れることができます。

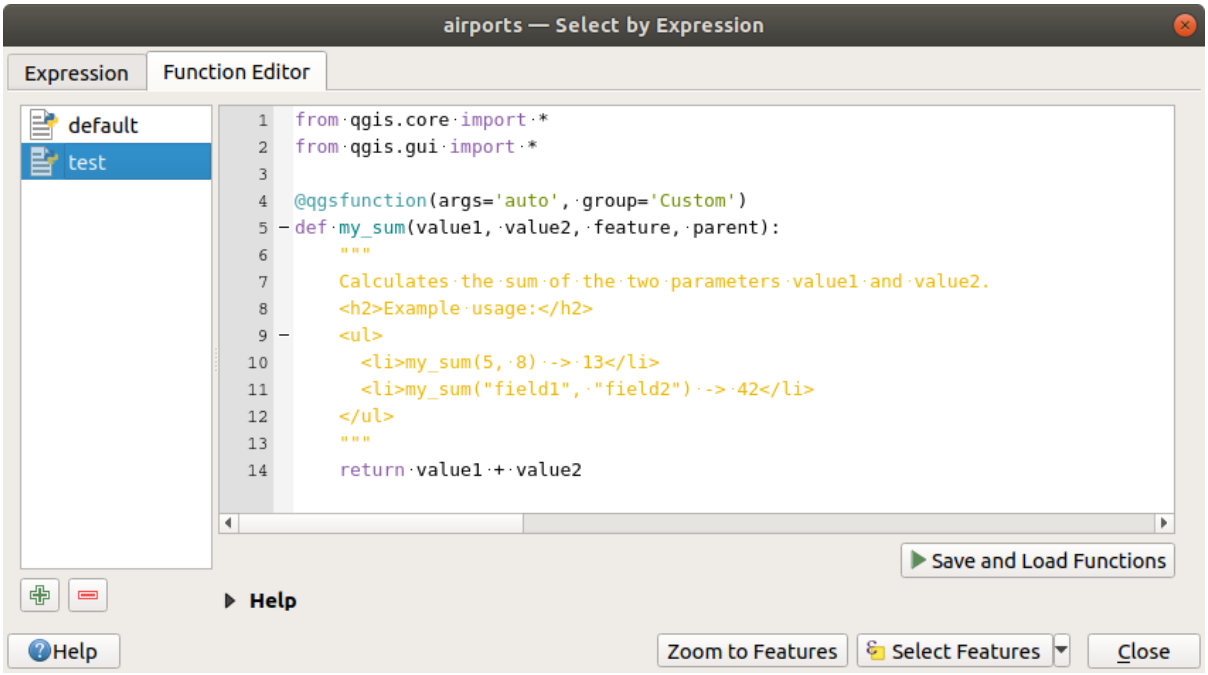


図 13.2: 関数エディタタブ

新しい関数を作成するには次のようにします。

1. 新規ファイル ボタンをクリックします。
2. フォームがポップアップするので、使用する名前を入力して *OK* をクリックします。

   入力した名前の新しいアイテムが **関数エディタ** タブの左側のパネルに追加されます。これは QGIS のテンプレートから生成された Python `.py` ファイルです。このファイルは現在有効になっている **ユーザープロファイル** のディレクトリ下の `/python/expressions` フォルダに格納されます。

3. 右側のパネルはファイルの内容を表示します。最初はテンプレートの Python スクリプトが表示されているので、このコードやヘルプを必要に応じて書き換えてください。
4. **関数の保存と読み込み** ボタンをクリックします。あなたの書いた関数が **式** タブの関数ツリーの中に追加されます。デフォルトでは Custom グループの中に追加されます。
5. 新しい関数を使いましょう。
6. 関数に改善が必要なときは、**関数エディタ** タブに戻って修正を加えた後、再度 **関数の保存と読み込み** ボタンをクリックして、ファイル内、つまりは任意の式タブで利用できるようにします。

カスタム Python 関数はユーザープロファイルのディレクトリ下に格納されます。これは QGIS が起動するたびに、現在のユーザープロファイルの下で定義されたすべての関数が自動的に読み込まれることを意味します。新しい関数は /python/expressions フォルダの中にのみ保存されて、プロジェクトファイルには保存されないことに注意してください。カスタム関数を使用したプロジェクトを共有するときは、/python/expressions フォルダの .py ファイルも共有する必要があります。

カスタム関数を削除するには：

1. **関数エディタ** タブを有効にします。
2. リストから削除したい関数を選択します。
3. 選択した関数ファイルを削除 ボタンを押します。関数がリストから削除され、対応する .py ファイルがユーザープロファイルフォルダから削除されます。

例

これは、2 つの値を操作する独自の my_sum 関数を作成する簡単な例です。

```
from qgis.core import *
from qgis.gui import *

@qgsfunction(args='auto', group='Custom')
def my_sum(value1, value2, feature, parent):
    """
    Calculates the sum of the two parameters value1 and value2.
    <h2>Example usage:</h2>
    <ul>
      <li>my_sum(5, 8) -> 13</li>
      <li>my_sum("field1", "field2") -> 42</li>
    </ul>
    """
    return value1 + value2
```

@qgsfunction デコレーターは次の引数を受け取ります:

- args: 引数の数。args='auto' を使用すると、必要な関数の引数の数は、Python で関数が定義されている引数の数（マイナス 2 - feature と parent ）から計算されます。args = -1 とすると、任意の数の引数を受け取ることができます。
- 引数 group は、式ダイアログで関数をリストするグループを示します。

- `usesgeometry=True` ：式が地物のジオメトリにアクセスする必要があるかどうか。デフォルトは `False` です。
- `handlesnull=True` ：式が NULL 値のカスタム処理を持つかどうか。 `False`（デフォルト）の場合、任意のパラメータが NULL であるときには常に即座に結果が NULL となります。
- `referenced_columns=[list]` ：この関数が必要とする属性名の配列。デフォルトは `[QgsFeatureRequest.ALL_ATTRIBUTES]` です。

この関数自体には次のような引数があります:

- 関数に渡したい任意の数と型のパラメータは、以下の引数よりも前に設定します。
- `feature`: 現在の地物
- `parent`: `QgsExpression` オブジェクト
- `context`: 最後の位置に `context` という引数がある場合、この変数には `QgsExpressionContext` オブジェクトが含まれ、式変数のような様々な追加情報へのアクセスを可能にします。例 `context.variable( 'layer_id' )`

上記の例の関数は、次のように式中で使うことができます。

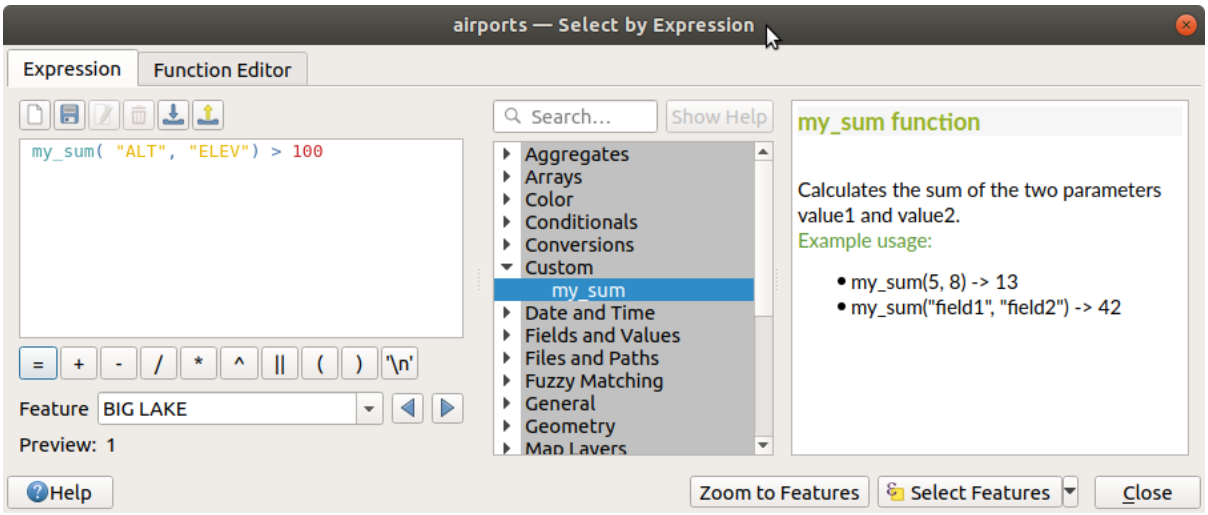


図 13.3: 式タブに追加されたカスタム関数

Python コードの作成に関する詳しい情報は PyQGIS-Developer-Cookbook で見つけることができます。

13.2 関数のリスト

QGIS で利用可能な関数、演算子、変数をカテゴリ別でまとめて以下に示します。

13.2.1 集計関数

このグループには、レイヤやフィールドの値を集計する関数が含まれます。

aggregate

別のレイヤの地物を使って計算された値の集計値を返します。

構文	aggregate(layer, aggregate, expression, [filter], [concatenator=''], [order_by]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - レイヤ名かレイヤ ID のどちらかを表す文字列 • **aggregate** - 計算する集計値に対応する文字列。有効なオプションは次のとおり： – count – count_distinct – count_missing – min – max – sum – mean – median – stdev – stdevsample – range – minority – majority – q1: 第 1 四分位数 – q3: 第 3 四分位数 – iqr: 四分位範囲 – min_length: 文字列長の最小値 – max_length: 文字列長の最大値 – concatenate: 文字列を連結文字で連結 – concatenate_unique: ユニークな文字列のみを連結文字で連結 – collect: マルチパートジオメトリに集約したものを生成 – array_agg: 集約値の配列を生成 • **expression** - 集約の対象とするサブ式またはフィールド名 • **filter** - （オプション）集約の計算に使用する地物を制限するフィルタ式。フィールドとジオメトリは結合されるレイヤの地物からのものです。ソースの地物には変数 @parent でアクセスできます。 • **concatenator** - `concatenate' および 'concatenate_unique' 集約の値を結合するために使用するオプションの文字列です。 • **order_by** -（オプション）集約値の計算に使用される地物を並べ替えるためのフィルタ式。フィールドとジオメトリは結合されるレイヤの地物からのものです。デフォルトでは、返される地物の順序は決まっていません。
例	• `aggregate(layer:='rail_stations',aggregate:='sum', expression:="passengers")` → rail_stations レイヤの passengers フィールドの値の合計 • `aggregate('rail_stations','sum', "passengers"/7)` → "passengers" フィールドを合計する前に 7 で割って、 "passengers" の日平均を計算する • `aggregate(layer:='rail_stations',aggregate:='sum', expression:="passengers",filter:="class">3)` → "class" 属性が 3 より大きい地物だけ、 "passengers" フィールドの値を合計する • `aggregate(layer:='rail_stations',aggregate:='concatenate', expression:="name", concatenator:=',')` → rail_stations レイヤの全地物の name フィールドをコンマ区切りで連結する • `aggregate(layer:='countries', aggregate:='max', expression:="code", filter:=intersects( $geometry, geometry(@parent) ) )` → 'countries' レイヤ上で交差している国の国コード

array_agg

属性値を集約し、配列として返します。

構文	array_agg(expression, [group_by], [filter], [order_by]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** -（オプション）集約計算のグループ化に使う式 • **filter** -（オプション）集約計算の地物フィルタに用いる式 • **order_by** -（オプション）集約計算の地物を並べ替えるための式。デフォルトでは、返される地物の順序は決まっていません。
例	• `array_agg("name",group_by:="state")` → state フィールドでグループ化された、name フィールドの値のリスト

collect

式で集約されたジオメトリをマルチパートジオメトリとして返します。

構文	collect(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集約するジオメトリ式 • **group_by** -（オプション）集約計算のグループ化に使う式 • **filter** -（オプション）集約計算の地物フィルタに用いる式
例	• `collect( $geometry )` → 集約されたジオメトリのマルチパートジオメトリ • `collect( centroid($geometry), group_by:="region", filter:= "use" = 'civilian' )` → region の値に基づいて civilian の地物の重心を集約する

concatenate

フィールドまたは式の値を区切り文字（デリミタ）で連結した文字列を返します。

構文	concatenate(expression, [group_by], [filter], [concatenator], [order_by]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式 • **concatenator** - （オプション）区切り文字列。デフォルトは空 • **order_by** - （オプション）集約計算の地物を並べ替えるための式。デフォルトでは、返される地物の順序は決まっていません。
例	• `concatenate("town_name",group_by:="state",concatenator:=',')` → state フィールドでグループ化された、 town_name のコンマ区切りのリスト

concatenate_unique

フィールドまたは式による文字列を重複を除外して区切り文字（デリミタ）で連結した文字列を返します。

構文	concatenate_unique(expression, [group_by], [filter], [concatenator], [order_by]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式 • **concatenator** - （オプション）区切り文字列。デフォルトは空 • **order_by** - （オプション）集約計算の地物を並べ替えるための式。デフォルトでは、返される地物の順序は決まっていません。
例	• `concatenate_unique("town_name",group_by:="state",concatenator:=', ')` → state フィールドでグループ化された、重複のない town_name のコンマ区切りのリスト

count

マッチする地物の数を返します。

構文	count(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `count("stations",group_by:="state")` → state フィールドでグループ化された stations の数

count_distinct

重複を除いた（distinct な）値の数を返します。

構文	count_distinct(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `count_distinct("stations",group_by:="state")` → state フィールドでグループ化された、重複のない stations の値の数

count_missing

欠損値（NULL）の数を返します

構文	count_missing(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `count_missing("stations",group_by:="state")` → state フィールドでグループ化された、stations の欠損値（NULL）の数

iqr

フィールドまたは式の値の四分位範囲（IQR）を返します。

構文	iqr(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `iqr("population",group_by:="state")` → state フィールドでグループ化された、population の値の四分位範囲

majority

フィールドまたは式の最頻値（最もよく出現する値）を返します。

構文	majority(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `majority("class",group_by:="state")` → state フィールドでグループ化された、最も頻繁に出現する class の値

max_length

フィールドまたは式の文字列の最大長を返します。

構文	max_length(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `max_length("town_name",group_by:="state")` → state フィールドでグループ化された town_name 文字列の最大長

maximum

フィールドまたは式の最大値を返します。

構文	maximum(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `maximum("population",group_by:="state")` → state フィールドでグループ化された、population の最大値

mean

フィールドまたは式の平均値を返します。

構文	mean(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `mean("population",group_by:="state")` → state フィールドでグループ化された、population の平均値

median

フィールドまたは式の中央値を返します。

構文	median(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `median("population",group_by:="state")` → state フィールドでグループ化された、 population の中央値

min_length

フィールドまたは式の文字列の最短長を返します。

構文	min_length(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `min_length("town_name",group_by:="state")` → state フィールドでグループ化された town_name の文字列の最短長

minimum

フィールドまたは式の最小値を返します。

構文	minimum(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `minimum("population",group_by:="state")` → state フィールドでグループ化された、population の最小値

minority

フィールドまたは式の最少出現値（出現が最も少ない値）を返します。

構文	minority(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `minority("class",group_by:="state")` → state フィールドでグループ化された、出現が最も稀な class の値

q1

フィールドまたは式の値の第１四分位数を返します。

構文	q1(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `q1("population",group_by:="state")` → state フィールドでグループ化された、population の値の第１四分位数

q3

フィールドまたは式の値の第３四分位数を返します。

構文	q3(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `q3("population",group_by:="state")` → state フィールドでグループ化された、population の値の第３四分位数

range

フィールドまたは式の値の範囲（最大値 - 最小値）を返します。

構文	range(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `range("population",group_by:="state")` → state フィールドでグループ化された、population の値の範囲

relation_aggregate

レイヤのリレーションからマッチする子地物を使用して計算された集計値を返します。

構文	relation_aggregate(relation, aggregate, expression, [concatenator=''], [order_by]) 記号 [] は、オプションの引数を意味します。
引数	• **relation** - リレーション ID を表す文字列 • **aggregate** - 計算する集計値に対応する文字列。有効なオプションは次のとおり： – count – count_distinct – count_missing – min – max – sum – mean – median – stdev – stdevsample – range – minority – majority – q1: 第 1 四分位数 – q3: 第 3 四分位数 – iqr: 四分位範囲 – min_length: 文字列長の最小値 – max_length: 文字列長の最大値 – concatenate: 文字列を連結文字で連結 – concatenate_unique: ユニークな文字列のみを連結文字で連結 – collect: マルチパートジオメトリに集約したものを生成 – array_agg: 集約値の配列を生成 • **expression** - 集約の対象とするサブ式またはフィールド名 • **concatenator** - （オプション） 'concatenate' 集約で値の結合に使用される文字 • **order_by** - （オプション）集約値の計算に使用される地物を並べ替えるための式。フィールドとジオメトリは結合されるレイヤの地物からのものです。デフォルトでは、返される地物の順序は決まっていません。
例	• `relation_aggregate(relation:='my_relation',aggregate:='mean', expression:="passengers")` → 'my_relation' リレーションを使用してマッチする子地物の passengers の値の平均値 • `relation_aggregate('my_relation','sum', "passengers"/7)` → 'my_relation' リレーションを使用してマッチする子地物の passengers フィールドを 7 で割った値の合計 • `relation_aggregate('my_relation','concatenate', "towns", concatenator:=',')` → 'my_relation' リレーションを使用してマッチする子地物の towns フィールドをコンマ区切りで連結したリスト • `relation_aggregate('my_relation','array_agg', "id")` → 'my_relation' リレーションを使用してマッチする子地物の id フィールドの配列

関連する項目: *1* **対多または多対多のリレーションの作成**

stdev

フィールドまたは式の標準偏差を返します。

構文	stdev(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `stdev("population",group_by:="state")` → state フィールドでグループ化された、population の値の標準偏差

sum

フィールドまたは式の合計値を返します。

構文	sum(expression, [group_by], [filter]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 集計する属性（サブ式） • **group_by** - （オプション）集約計算のグループ化に使う式 • **filter** - （オプション）集約計算の地物フィルタに用いる式
例	• `sum("population",group_by:="state")` → state フィールドでグループ化された、population の合計値

13.2.2 配列関数

このグループには、配列（リストデータ構造とも呼ばれます）の作成と操作のための関数が含まれています。配列では、値の順序こそが重要です。これは、キーと値のペアの順序は関係なく、値はキーによって識別される **「マップ型」のデータ構造** とは異なっています。

array

パラメータとして渡されたすべての値を含む配列を返します。

構文	array(value1, value2, ...)
引数	• **value** - 値
例	• `array(2,10)` → [2, 10] • `array(2,10)[0]` → 2

array_all

配列が与えられた配列のすべての値を含む場合に TRUE を返します。

構文	array_all(array_a, array_b)
引数	• **array_a** - 配列 • **array_b** - 値を検索する配列
例	• `array_all(array(1,2,3),array(2,3))` → TRUE • `array_all(array(1,2,3),array(1,2,4))` → FALSE

array_append

与えられた値が最後に追加された配列を返します。

構文	array_append(array, value)
引数	• **array** - 配列 • **value** - 追加する値
例	• `array_append(array(1,2,3),4)` → [1, 2, 3, 4]

array_cat

与えられたすべての配列を連結して、一つの配列として返します。

構文	array_cat(array1, array2, ...)
引数	• **array** - 配列
例	• `array_cat(array(1,2),array(2,3))` → [1, 2, 2, 3]

array_contains

配列が与えられた値を含む場合に TRUE を返します。

構文	array_contains(array, value)
引数	• **array** - 配列 • **value** - 検索する値
例	• `array_contains(array(1,2,3),2)` → TRUE

array_count

与えられた値が配列内に存在する数をカウントします。

構文	array_count(array, value)
引数	• **array** - 配列 • **value** - カウントする値
例	• `array_count(array('a', 'b', 'c', 'b'), 'b')` → 2

array_distinct

与えられた配列の値から、相異なる値を含む配列を返します。

構文	array_distinct(array)
引数	• **array** - 配列
例	• `array_distinct(array(1,2,3,2,1))` → [1, 2, 3]

array_filter

式が true と判定した要素だけを含む配列を返します。

構文	array_filter(array, expression, [limit=0]) 記号 [] は、オプションの引数を意味します。
引数	• **array** - 配列 • **expression** - それぞれの要素に対して評価する式。変数 *@element* が各要素を表します。 • **limit** - 戻り値の要素の最大数。0 の場合はすべてを返します。
例	• `array_filter(array(1,2,3),@element < 3)` → [1, 2] • `array_filter(array(1,2,3),@element < 3, 1)` → [1]

array_find

ある値の配列内における最小インデックス（最初は 0 から数える）を返します。値が配列内で見つからない場合には -1 を返します。

構文	array_find(array, value)
引数	• **array** - 配列 • **value** - 検索する値
例	• `array_find(array('a', 'b', 'c'), 'b')` → 1 • `array_find(array('a', 'b', 'c', 'b'), 'b')` → 1

array_first

配列の最初の値を返します。

構文	array_first(array)
引数	• **array** - 配列
例	• `array_first(array('a','b','c'))` → 'a'

array_foreach

配列の各要素を与えられた式で評価した結果を配列として返します。

構文	array_foreach(array, expression)
引数	• **array** - 配列 • **expression** - それぞれの要素に対して評価する式。変数 *@element* が各要素を表します。
例	• `array_foreach(array('a','b','c'),upper(@element))` → ['A', 'B', 'C'] • `array_foreach(array(1,2,3),@element + 10)` → [11, 12, 13]

array_get

配列の N 番目の値（最初は 0 から数える）、または最後から -N 番目の値（最後は-1 から数える）を返します。

構文	array_get(array, pos)
引数	• **array** - 配列 • **pos** - 取り出すインデックス（最初は 0）
例	• `array_get(array('a','b','c'),1)` → 'b' • `array_get(array('a','b','c'),-1)` → 'c'

ヒント: 配列から値を取得するには **インデックス演算子** *([])* も使用できます。

array_insert

配列の指定された位置に値が挿入された配列を返します。

構文	array_insert(array, pos, value)
引数	• **array** - 配列 • **pos** - 挿入する位置（最初は 0） • **value** - 追加する値
例	• `array_insert(array(1,2,3),1,100)` → [1, 100, 2, 3]

array_intersect

array1 の要素が array2 に少なくとも 1 つ存在する場合に TRUE を返します。

構文	array_intersect(array1, array2)
引数	• **array1** - 配列 • **array2** - 別の配列
例	• `array_intersect(array(1,2,3,4),array(4,0,2,5))` → TRUE

array_last

配列の最後の値を返します。

構文	array_last(array)
引数	• **array** - 配列
例	• `array_last(array('a','b','c'))` → 'c'

array_length

配列の要素の数を返します。

構文	array_length(array)
引数	• **array** - 配列
例	• `array_length(array(1,2,3))` → 3

array_majority

配列内で最も頻出する値を返します。

構文	array_majority(array, [option='all']) 記号 [] は、オプションの引数を意味します。
引数	• **array** - 配列 • **option='all'** - 戻り値の処理を指定する文字列。有効なオプションは次のとおり： – all: デフォルトの設定で、最頻値すべてを配列形式で返します。 – any: 最頻値の一つを返します。 – median: 最頻値の中央値を返します。非数値は無視されます。 – real_majority: 配列サイズの半数よりも多い値を返します。
例	• `array_majority(array(0,1,42,42,43), 'all')` → [42] • `array_majority(array(0,1,42,42,43,1), 'all')` → [42, 1] • `array_majority(array(0,1,42,42,43,1), 'any')` → 1 か 42 • `array_majority(array(0,1,1,2,2), 'median')` → 1.5 • `array_majority(array(0,1,42,42,43), 'real_majority')` → NULL • `array_majority(array(0,1,42,42,43,42), 'real_majority')` → NULL • `array_majority(array(0,1,42,42,43,42,42), 'real_majority')` → 42

array_max

配列の最大値を返します。

構文	array_max(array)
引数	• **array** - 配列
例	• `array_max(array(0,42,4,2))` → 42

array_mean

配列の算術平均値を返します。配列内の非数値は無視されます。

構文	array_mean(array)
引数	• **array** - 配列
例	• `array_mean(array(0,1,7,66.6,135.4))` → 42 • `array_mean(array(0,84,'a','b','c'))` → 42

array_median

配列の中央値を返します。配列内の非数値は無視されます。

構文	array_median(array)
引数	• **array** - 配列
例	• `array_median(array(0,1,42,42,43))` → 42 • `array_median(array(0,1,2,42,'a','b'))` → 1.5

array_min

配列の最小値を返します。

構文	array_min(array)
引数	• **array** - 配列
例	• `array_min(array(43,42,54))` → 42

array_minority

配列内で最も稀な値を返します。

構文	array_minority(array, [option='all']) 記号 [] は、オプションの引数を意味します。
引数	• **array** - 配列 • **option='all'** - 戻り値の処理を指定する文字列。有効なオプションは次のとおり： – all: デフォルトの設定で、最稀値すべてを配列形式で返します。 – any: 最稀値の一つを返します。 – median: 最稀値の中央値を返します。非数値は無視されます。 – real_minority: 配列サイズの半数以下の値を返します。
例	• `array_minority(array(0,42,42), 'all')` → [0] • `array_minority(array(0,1,42,42), 'all')` → [0, 1] • `array_minority(array(0,1,42,42,43,1), 'any')` → 0 か 43 • `array_minority(array(1,2,3,3), 'median')` → 1.5 • `array_minority(array(0,1,42,42,43), 'real_minority')` → [42, 43, 0, 1] • `array_minority(array(0,1,42,42,43,42), 'real_minority')` → [42, 43, 0, 1] • `array_minority(array(0,1,42,42,43,42,42), 'real_minority')` → [43, 0, 1]

array_prepend

与えられた値が先頭に追加された配列を返します。

構文	array_prepend(array, value)
引数	• **array** - 配列 • **value** - 追加する値
例	• `array_prepend(array(1,2,3),0)` → [0, 1, 2, 3]

array_prioritize

ある配列を別の配列で指定された順序でソートしたものを返します。第 1 引数の配列にはあり、第 2 引数の配列には無い値は、結果の配列の末尾に追加されます。

構文	array_prioritize(array, array_prioritize)
引数	• **array** - 配列 • **array_prioritize** - 値の優先順で並べられた配列
例	• `array_prioritize(array(1, 8, 2, 5), array(5, 4, 2, 1, 3, 8))` → [5, 2, 1, 8] • `array_prioritize(array(5, 4, 2, 1, 3, 8), array(1, 8, 6, 5))` → [1, 8, 5, 4, 2, 3]

array_remove_all

配列から指定された値がすべて削除された配列を返します。

構文	array_remove_all(array, value)
引数	• **array** - 配列 • **value** - 削除する値
例	• `array_remove_all(array('a','b','c','b'),'b')` → ['a', 'c']

array_remove_at

与えられたインデックスの項目を取り除いた配列を返します。正のインデックス（最初の要素は 0）および負のインデックス（最後の-N 番目の値、最後の要素は-1）をサポートします。

構文	array_remove_at(array, pos)
引数	• **array** - 配列 • **pos** - 削除する位置（最初は 0）
例	• `array_remove_at(array(1, 2, 3), 1)` → [1, 3] • `array_remove_at(array(1, 2, 3), -1)` → [1, 2]

array_replace

配列を指定した別の値、配列またはマップ型オブジェクトで置換した配列を返します。

値・配列バージョン

指定された値または値の配列を、別の値または値の配列で置換した配列を返します。

構文	array_replace(array, before, after)
引数	• **array** - 入力配列 • **before** - 置換される値または配列 • **after** - 置き換えに使用する値または配列
例	• `array_replace(array('QGIS','SHOULD','ROCK'),'SHOULD','DOES')` → ['QGIS', 'DOES', 'ROCK'] • `array_replace(array(3,2,1),array(1,2,3),array(7,8,9))` → [9, 8, 7] • `array_replace(array('Q','G','I','S'),array('Q','S'),'-')` → ['-', 'G', 'I', '-']

マップ型バージョン

指定されたマップ型のキーを対応する値で置き換えた配列を返します。

構文	array_replace(array, map)
引数	• **array** - 入力配列 • **map** - キーと値のペアを含むマップ型オブジェクト
例	• `array_replace(array('APP', 'SHOULD', 'ROCK'),map('APP','QGIS', 'SHOULD','DOES'))` → ['QGIS', 'DOES', 'ROCK']

array_reverse

逆順にした配列を返します。

構文	array_reverse(array)
引数	• **array** - 配列
例	• `array_reverse(array(2,4,0,10))` → [10, 0, 4, 2]

array_slice

配列の一部を返します。スライスは、start_pos と end_pos の引数で定義されます。

構文	array_slice(array, start_pos, end_pos)
引数	• **array** - 配列 • **start_pos** - スライスの開始位置のインデックス（最初は 0）。start_pos インデックスの値はスライス内に含まれます。start_pos を負の値とした場合、インデックスをリストの最後から数えます（最後の要素は-1）。 • **end_pos** - スライスの終了位置のインデックス（最初は 0）。end_pos インデックスの値はスライス内に含まれます。end_pos を負の値とした場合、インデックスをリストの最後から数えます（最後の要素は-1）。
例	• `array_slice(array(1,2,3,4,5),0,3)` → [1, 2, 3, 4] • `array_slice(array(1,2,3,4,5),0,-1)` → [1, 2, 3, 4, 5] • `array_slice(array(1,2,3,4,5),-5,-1)` → [1, 2, 3, 4, 5] • `array_slice(array(1,2,3,4,5),0,0)` → [1] • `array_slice(array(1,2,3,4,5),-2,-1)` → [4, 5] • `array_slice(array(1,2,3,4,5),-1,-1)` → [5] • `array_slice(array('Dufour','Valmiera','Chugiak','Brighton'),1,2)` → ['Valmiera', 'Chugiak'] • `array_slice(array('Dufour','Valmiera','Chugiak','Brighton'),-2,-1)` → ['Chugiak', 'Brighton']

array_sort

指定された配列の要素をソートして返します。

構文	array_sort(array, [ascending=true]) 記号 [] は、オプションの引数を意味します。
引数	• **array** - 配列 • **ascending** - このパラメータを false とすると、配列を降順でソートする
例	• `array_sort(array(3,2,1))` → [1, 2, 3]

array_sum

配列の合計値を返します。配列内の非数値は無視されます。

構文	array_sum(array)
引数	• **array** - 配列
例	• `array_sum(array(0,1,39.4,1.6,'a'))` → 42.0

array_to_string

配列の要素を区切り文字で連結した文字列を返します。空の値を置き換える文字列を指定できます。

構文	array_to_string(array, [delimiter=','], [empty_value='']) 記号 [] は、オプションの引数を意味します。
引数	• **array** - 入力配列 • **delimiter** - 連結する配列要素を区切るために使用する区切り文字 • **empty_value** - （オプション）空のマッチ（長さゼロ）の代わりに使用する文字列
例	• `array_to_string(array('1','2','3'))` → '1,2,3' • `array_to_string(array(1,2,3),'-')` → '1-2-3' • `array_to_string(array('1','','3'),',','0')` → '1,0,3'

generate_series

連続した数字からなる配列を作ります。

構文	generate_series(start, stop, [step=1]) 記号 [] は、オプションの引数を意味します。
引数	• **start** - シーケンスの最初の値 • **stop** - シーケンスの最後の値 • **step** - 増分値
例	• `generate_series(1,5)` → [1, 2, 3, 4, 5] • `generate_series(5,1,-1)` → [5, 4, 3, 2, 1]

geometries_to_array

ジオメトリをより単純なジオメトリに分割し、配列にします。

構文	geometries_to_array(geometry)
引数	• **geometry** - 入力ジオメトリ
例	• `geometries_to_array(geom_from_wkt('GeometryCollection (Polygon ((5 8, 4 1, 3 2, 5 8)),LineString (3 2, 4 2))'))` → ポリゴンとラインジオメトリの配列 • `geom_to_wkt(geometries_to_array(geom_from_wkt('GeometryCollection (Polygon ((5 8, 4 1, 3 2, 5 8)),LineString (3 2, 4 2))'))[0])` → 'Polygon ((5 8, 4 1, 3 2, 5 8))' • `geometries_to_array(geom_from_wkt('MULTIPOLYGON(((5 5,0 0,0 10,5 5)),((5 5,10 10,10 0,5 5))'))` → 二つのポリゴンジオメトリの配列

regexp_matches

文字列に対する正規表現でグループが現れた順に、キャプチャグループによってキャプチャされた文字列すべての配列を返します。

構文	regexp_matches(string, regex, [empty_value='']) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 正規表現でグループをキャプチャする対象の文字列 • **regex** - グループをキャプチャするために使用する正規表現 • **empty_value** - （オプション）空のマッチ（長さゼロ）の代わりに使用する文字列
例	• `regexp_matches('QGIS=>rocks','(.*)=>(.*)')` → ['QGIS', 'rocks'] • `regexp_matches('key=>','(.*)=>(.*)','empty value')` → ['key', 'empty value']

string_to_array

文字列を指定された区切り文字で配列に分割します。空の値の代わりの文字列を指定するオプションがあります。

構文	string_to_array(string, [delimiter=','], [empty_value='']) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 入力文字列 • **delimiter** - 入力文字列を分割するための区切り文字 • **empty_value** - （オプション）空のマッチ（長さゼロ）の代わりに使用する文字列
例	• `string_to_array('1,2,3',',')` → ['1', '2', '3'] • `string_to_array('1,,3',',','0')` → ['1', '0', '3']

13.2.3 色関数

このグループには、色を操作するための関数が含まれています。

color_cmyk

シアン、マゼンタ、イエロー、ブラック成分に基づいて、色を表す文字列を返します。

構文	color_cmyk(cyan, magenta, yellow, black)
引数	• **cyan** - 色のシアン成分。0 から 100 までのパーセンテージの整数値 • **magenta** - 色のマゼンタ成分。0 から 100 までのパーセンテージの整数値 • **yellow** - 色のイエロー成分。0 から 100 までのパーセンテージの整数値 • **black** - 色のブラック成分。0 から 100 までのパーセンテージの整数値
例	• `color_cmyk(100,50,0,10)` → '0,115,230'

color_cmyka

シアン、マゼンタ、イエロー、ブラックおよびアルファ（透明度）成分に基づいて、色を表す文字列を返します。

構文	color_cmyka(cyan, magenta, yellow, black, alpha)
引数	• **cyan** - 色のシアン成分。0 から 100 までのパーセンテージの整数値 • **magenta** - 色のマゼンタ成分。0 から 100 までのパーセンテージの整数値 • **yellow** - 色のイエロー成分。0 から 100 までのパーセンテージの整数値 • **black** - 色のブラック成分。0 から 100 までのパーセンテージの整数値 • **alpha** - 色のアルファ成分。0（完全に透明）から 255（不透明）までの整数値
例	• `color_cmyka(100,50,0,10,200)` → '0,115,230,200'

color_grayscale_average

指定された色にグレースケールフィルタを適用した色を表す文字列を返します。

構文	color_grayscale_average(color)
引数	• **color** - 色を表す文字列
例	• `color_grayscale_average('255,100,50')` → '135,135,135,255'

color_hsl

色相、彩度、輝度属性に基づいて、色を表す文字列を返します。

構文	color_hsl(hue, saturation, lightness)
引数	• **hue** - 色相。0 から 360 までの整数値 • **saturation** - 彩度。0 から 100 までのパーセンテージの整数値 • **lightness** - 輝度。0 から 100 までのパーセンテージの整数値
例	• `color_hsl(100,50,70)` → '166,217,140'

color_hsla

色相、彩度、輝度およびアルファ（透明度）属性に基づいて、色を表す文字列を返します。

構文	color_hsla(hue, saturation, lightness, alpha)
引数	• **hue** - 色相。0 から 360 までの整数値 • **saturation** - 彩度。0 から 100 までのパーセンテージの整数値 • **lightness** - 輝度。0 から 100 までのパーセンテージの整数値 • **alpha** - 色のアルファ成分。0（完全に透明）から 255（不透明）までの整数値
例	• `color_hsla(100,50,70,200)` → '166,217,140,200'

color_hsv

色相、彩度、明度属性に基づいて、色を表す文字列を返します。

構文	color_hsv(hue, saturation, value)
引数	• **hue** - 色相。0 から 360 までの整数値 • **saturation** - 彩度。0 から 100 までのパーセンテージの整数値 • **value** - 明度。0 から 100 までのパーセンテージの整数値
例	• `color_hsv(40,100,100)` → '255,170,0'

color_hsva

色相、彩度、明度およびアルファ（透明度）属性に基づいて、色を表す文字列を返します。

構文	color_hsva(hue, saturation, value, alpha)
引数	• **hue** - 色相。0 から 360 までの整数値 • **saturation** - 彩度。0 から 100 までのパーセンテージの整数値 • **value** - 明度。0 から 100 までのパーセンテージの整数値 • **alpha** - 色のアルファ成分。0（完全に透明）から 255（不透明）までの整数値
例	• `color_hsva(40,100,100,200)` → '255,170,0,200'

color_mix_rgb

2 つの指定された色の赤、緑、青、アルファ値を、与えられた比率に基づいて混合させた色を表す文字列を返します。

構文	color_mix_rgb(color1, color2, ratio)
引数	• **color1** - 色を表す文字列 • **color2** - 色を表す文字列 • **ratio** - 混合比率
例	• `color_mix_rgb('0,0,0','255,255,255',0.5)` → '127,127,127,255'

color_part

色を表す文字列から、赤色成分やアルファ成分など特定の成分を返します。

構文	color_part(color, component)
引数	• **color** - 色を表す文字列 • **component** - 戻り値となる、色の成分に対応する文字列。有効なオプションは次のとおり： – red: RGB の赤色成分（0-255） – green: RGB の緑色成分（0-255） – blue: RGB の青色成分（0-255） – alpha: アルファ（透明度）値（0-255） – hue: HSV の色相（0-360） – saturation: HSV の彩度（0-100） – value: HSV の明度（0-100） – hsl_hue: HSL の色相（0-360） – hsl_saturation: HSL の彩度（0-100） – lightness: HSL の輝度（0-100） – cyan: CMYK のシアン成分（0-100） – magenta: CMYK のマゼンタ成分（0-100） – yellow: CMYK のイエロー成分（0-100） – black: CMYK のブラック成分（0-100）
例	• `color_part('200,10,30','green')` → 10

color_rgb

赤、緑、青成分に基づいて、色を表す文字列を返します。

構文	color_rgb(red, green, blue)
引数	• **red** - 赤色成分。0 から 255 までの整数値 • **green** - 緑色成分。0 から 255 までの整数値 • **blue** - 青色成分。0 から 255 までの整数値
例	• `color_rgb(255,127,0)` → '255,127,0'

color_rgba

赤、緑、青およびアルファ（透明度）成分に基づいて、色を表す文字列を返します。

構文	color_rgba(red, green, blue, alpha)
引数	• **red** - 赤色成分。0 から 255 までの整数値 • **green** - 緑色成分。0 から 255 までの整数値 • **blue** - 青色成分。0 から 255 までの整数値 • **alpha** - 色のアルファ成分。0（完全に透明）から 255（不透明）までの整数値
例	• `color_rgba(255,127,0,200)` → '255,127,0,200'

create_ramp

色文字列とステップのマップ型オブジェクトで定義された、グラデーションランプを返します。

構文	create_ramp(map, [discrete=false]) 記号 [] は、オプションの引数を意味します。
引数	• **map** - 色文字列とステップのマップ型オブジェクト • **discrete** - 離散的なカラーマップを作る場合、このパラメータを true に設定する
例	• `ramp_color(create_ramp(map(0,'0,0,0',1,'255,0,0')),1)` → '255,0,0,255'

darker

指定した色を基準に、より暗い（または明るい）色の色文字列を返します。

構文	darker(color, factor)
引数	• **color** - 色を表す文字列 • **factor** - 暗化係数に相当する整数 – 係数が 100 よりも大きい場合には、この関数はより暗い色を返します（例えば、係数を 200 とすると、輝度が半分の色が返されます）。 – 係数が 100 よりも小さい場合には、戻り値の色はより明るくなります。ただし、この目的では lighter() 関数を使用することを推奨します。 – 係数が 0 または負の値の場合には、戻り値は未定義です。
例	• `darker('200,10,30', 200)` → '100,5,15,255'

参考： *lighter*

lighter

指定した色を基準に、より明るい（または暗い）色の色文字列を返します。

構文	lighter(color, factor)
引数	• **color** - 色を表す文字列 • **factor** - 明化係数に相当する整数 – 係数が 100 よりも大きい場合には、この関数はより明るい色を返します（例えば、係数を 150 とすると、50%明るい色が返されます）。 – 係数が 100 よりも小さい場合には、戻り値の色はより暗くなります。ただし、この目的では darker() 関数を使用することを推奨します。 – 係数が 0 または負の値の場合には、戻り値は未定義です。
例	• `lighter('200,10,30', 200)` → '255,158,168,255'

参考： *darker*

project_color

プロジェクトのカラースキーマから色を返します。

構文	project_color(name)
引数	• **name** - 色の名前
例	• `project_color('Logo color')` → '20,140,50'

参考: **プロジェクトの色の設定**

ramp_color

カラーランプから、色を表す文字列を返します。

保存済みカラーランプバージョン

保存済みカラーランプから、色を表す文字列を返します。

構文	ramp_color(ramp_name, value)
引数	• **ramp_name** - カラーランプの名前の文字列。例： 'Spectral' • **value** - ランプ上で色を取得する位置。0 と 1 の間の実数値
例	• `ramp_color('Spectral',0.3)` → '253,190,115,255'

注釈: 利用可能なカラーランプは QGIS のインストール環境によって異なります。異なるインストール環境に QGIS プロジェクトを移動した場合、この関数は期待した結果が得られない場合があります。

式によって定義されたカラーランプバージョン

式によって定義されたカラーランプから、色を表す文字列を返します。

構文	ramp_color(ramp, value)
引数	• **ramp** - カラーランプ • **value** - ランプ上で色を取得する位置。0 と 1 の間の実数値
例	• `ramp_color(create_ramp(map(0,'0,0,0',1,'255,0,0')),1)` → '255,0,0,255'

参考： **カラーランプの設定** 、 **カラーランプのドロップダウンショートカット**

set_color_part

色を表す文字列に対して、赤色成分やアルファ成分など特定の色成分を設定します。

構文	set_color_part(color, component, value)
引数	• **color** - 色を表す文字列 • **component** - 設定する色の成分に対応する文字列。有効なオプションは次のとおり： – red: RGB の赤色成分（0-255） – green: RGB の緑色成分（0-255） – blue: RGB の青色成分（0-255） – alpha: アルファ（透明度）値（0-255） – hue: HSV の色相（0-360） – saturation: HSV の彩度（0-100） – value: HSV の明度（0-100） – hsl_hue: HSL の色相（0-360） – hsl_saturation: HSL の彩度（0-100） – lightness: HSL の輝度（0-100） – cyan: CMYK のシアン成分（0-100） – magenta: CMYK のマゼンタ成分（0-100） – yellow: CMYK のイエロー成分（0-100） – black: CMYK のブラック成分（0-100） • **value** - 色成分の新しい値。上記の値の範囲に従う
例	• `set_color_part('200,10,30','green',50)` → '200,50,30,255'

13.2.4 条件関数

このグループには、式の中で条件分けに使う関数があります。

CASE

CASE は、一連の条件を評価し、条件が最初に満たされる結果を返します。条件は上から順に評価され、ある条件が true であれば評価を停止し、対応する結果が返されます。どの条件も true でない場合は、ELSE 節の値が返されます。ELSE 節が無く、いずれの条件も満たさない場合には、NULL が返されます。

CASE

WHEN *condition* THEN *result*

[⋯n]

[ELSE *result*]

END

記号 [] は、オプションの引数を意味します。

引数	• **WHEN condition** - 評価される条件式 • **THEN result** - *condition* 式が true の場合、 *result* 式が評価され、返されます。 • **ELSE result** - 上にある条件式がどれも true と評価されない場合には、 *result* 式が評価され、返されます。
例	• `CASE WHEN "name" IS NULL THEN 'None' END` → "name" フィールドが NULL の場合には、文字列 'None' を返す • `CASE WHEN $area > 10000 THEN 'Big property' WHEN $area > 5000 THEN 'Medium property' ELSE 'Small property' END` → 面積が 10000 よりも大きい場合には文字列 'Big property' を、面積が 5000 から 10000 の場合には文字列 'Medium property' を、それ以外の場合には文字列 'Small property' を返す

coalesce

式リストから最初の非 NULL 値を返します。

この関数は、任意の数の引数をとることができます。

構文	coalesce(expression1, expression2, ...)
引数	• **expression** - 型に関わらず、有効な式または値。
例	• `coalesce(NULL, 2)` → 2 • `coalesce(NULL, 2, 3)` → 2 • `coalesce(7, NULL, 3*2)` → 7 • `coalesce("fieldA", "fallbackField", 'ERROR')` → fieldA が非 NULL の場合には fieldA の値。fieldA が NULL の場合は fallbackField の値。両方とも NULL の場合には、文字列 'ERROR'

if

条件をテストし、それに応じて異なる結果を返します。

構文	if(condition, result_when_true, result_when_false)
引数	• **condition** - チェックする条件 • **result_when_true** - 条件が true か、あるいは false に変換できない値である場合に返される結果 • **result_when_false** - 条件が false であるか、0 や長さゼロの文字列 '' といった false に変換される値である場合に返される結果。NULL も false に変換されます。
例	• `if( 1+1=2, 'Yes', 'No' )` → 'Yes' • `if( 1+1=3, 'Yes', 'No' )` → 'No' • `if( 5 > 3, 1, 0)` → 1 • `if( '', 'It is true (not empty)', 'It is false (empty)' )` → 'It is false (empty)' • `if( ' ', 'It is true (not empty)', 'It is false (empty)' )` → 'It is true (not empty)' • `if( 0, 'One', 'Zero' )` → 'Zero' • `if( 10, 'One', 'Zero' )` → 'One'

nullif

value1 が value2 と等しい場合には NULL を返し、そうでない場合には value1 を返します。この関数は、条件によって値を NULL に置き換える場合に便利です。

構文	nullif(value1, value2)
引数	• **value1** - そのまま使われるか、NULL に置き換えられる値 • **value2** - NULL 置換の基準になる値
例	• `nullif('(none)', '(none)')` → NULL • `nullif('text', '(none)')` → 'text' • `nullif("name", '')` → name が空文字列（または NULL）の場合は NULL、そうでない場合は name

regexp_match

unicode 文字列内で正規表現にマッチする最初の位置を返します。部分文字列が見つからない場合、0 を返します。

構文	regexp_match(input_string, regex)
引数	• **input_string** - 正規表現に対してテストする文字列 • **regex** - テストする正規表現。バックスラッシュは二重にエスケープする必要があります（たとえば、空白文字に一致させる場合は "\\s" 、単語の境界に一致させる場合は "\\b" ）
例	• `regexp_match('QGIS ROCKS','\\sROCKS')` → 5 • `regexp_match('Budač','udač\\b')` → 2

try

式の評価を試し、エラーがなければその値を返します。式がエラーとなる場合には、指定された代替値を返します。指定値がない場合は NULL を返します。

構文	try(expression, [alternative]) 記号 [] は、オプションの引数を意味します。
引数	• **expression** - 評価したい式 • **alternative** - 式がエラーを返す場合に返したい値
例	• `try( to_int( '1' ), 0 )` → 1 • `try( to_int( 'a' ), 0 )` → 0 • `try( to_date( 'invalid_date' ) )` → NULL

13.2.5 変換関数

このグループには、あるデータ型を他の型に変換する関数が含まれます（例えば文字列型と整数型の変換、文字列型とバイナリ型の変換、文字列から日付への変換など）。

from_base64

Base64 エンコードの文字列をバイナリ値にデコードします。

構文	from_base64(string)
引数	• **string** - デコードする文字列
例	• `from_base64('UUdJUw==')` → 'QGIS'

hash

指定された方法で文字列からハッシュを作ります。1 バイト（8 ビット）は 16 進数 2 文字で表されるので、'md4' （16 bytes）では 16 * 2 = 32 文字に、 'keccak_512' （64 bytes）では 64 * 2 = 128 文字になります。

構文	hash(string, method)
引数	• **string** - ハッシュ化する文字列 • **method** - ハッシュ化方法： 'md4', 'md5', 'sha1', 'sha224', 'sha384', 'sha512', 'sha3_224', 'sha3_256', 'sha3_384', 'sha3_512', 'keccak_224', 'keccak_256', 'keccak_384', 'keccak_512'
例	• `hash('QGIS', 'md4')` → 'c0fc71c241cdebb6e888cbac0e2b68eb' • `hash('QGIS', 'md5')` → '57470aaa9e22adaefac7f5f342f1c6da' • `hash('QGIS', 'sha1')` → 'f87cfb2b74cdd5867db913237024e7001e62b114' • `hash('QGIS', 'sha224')` → '4093a619ada631c770f44bc643ead18fb393b93d6a6af1861fcfece0' • `hash('QGIS', 'sha256')` → 'eb045cba7a797aaa06ac58830846e40c8e8c780bc0676d3393605fae50c053 • `hash('QGIS', 'sha384')` → '91c1de038cc3d09fdd512e99f9dd9922efadc39ed21d3922e69a4305cc2550 • `hash('QGIS', 'sha512')` → 'c2c092f2ab743bf8edbeb6d028a745f30fc720408465ed369421f0a4e20fa5e • `hash('QGIS', 'sha3_224')` → '467f49a5039e7280d5d42fd433e80d203439e338eaabd701f0d6c17d' • `hash('QGIS', 'sha3_256')` → '540f7354b6b8a6e735f2845250f15f4f3ba4f666c55574d9e9354575de0e • `hash('QGIS', 'sha3_384')` → '96052da1e77679e9a65f60d7ead961b287977823144786386eb43647b0 • `hash('QGIS', 'sha3_512')` → '900d079dc69761da113980253aa8ac0414a8bd6d09879a916228f87437 • `hash('QGIS', 'keccak_224')` → '5b0ce6acef8b0a121d4ac4f3eaa8503c799ad4e26a3392d1fb201478' • `hash('QGIS', 'keccak_256')` → '991c520aa6815392de24087f61b2ae0fd56abbfeee4a8ca019c1011d32 • `hash('QGIS', 'keccak_384')` → 'c57a3aed9d856fa04e5eeee9b62b6e027cca81ba574116d3cc1f0d48a1

md5

文字列から md5 ハッシュを作ります。

構文	md5(string)
引数	• **string** - ハッシュ化する文字列
例	• `md5('QGIS')` → '57470aaa9e22adaefac7f5f342f1c6da'

sha256

文字列から sha256 ハッシュを作ります。

構文	sha256(string)
引数	• **string** - ハッシュ化する文字列
例	• `sha256('QGIS')` → 'eb045cba7a797aaa06ac58830846e40c8e8c780bc0676d3393605fae50c05309'

to_base64

バイナリ値を Base64 エンコードを使用して文字列にエンコードします。

構文	to_base64(value)
引数	• **value** - エンコードするバイナリ値
例	• `to_base64('QGIS')` → 'UUdJUw=='

to_date

文字列を日付型オブジェクトに変換します。オプションとして、文字列をパースするためのフォーマット文字列を指定できます。フォーマットに関するドキュメントは、 QDate::fromString または format_date 関数のドキュメントを参照してください。デフォルトでは現在の QGIS ユーザーのロケールを使用します。

構文	to_date(string, [format], [language]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 日付を表す文字列 • **format** - 文字列を日付型に変換するために使用するフォーマット • **language** - 日付型に変換する文字列の言語（2 文字または 3 文字の小文字、ISO 639 言語名コード）。デフォルトでは現在の QGIS ユーザーのロケールを使用します
例	• `to_date('2012-05-04')` → 2012-05-04 • `to_date('June 29, 2019','MMMM d, yyyy')` → 2019-06-29 （現在のロケールが 6 番目の月名に 'June' を使用する言語の場合の結果。そうでない場合はエラーが発生します） • `to_date('29 juin, 2019','d MMMM, yyyy','fr')` → 2019-06-29

to_datetime

文字列を日付時刻型オブジェクトに変換します。オプションとして、文字列をパースするためのフォーマット文字列を指定できます。フォーマットに関するドキュメントは、QDate::fromString 、QTime::fromString または format_date 関数のドキュメントを参照してください。デフォルトでは現在の QGIS ユーザーのロケールを使用します。

構文	to_datetime(string, [format], [language]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 日付時刻を表す文字列 • **format** - 文字列を日付時刻型に変換するために使用するフォーマット • **language** - 日付時刻型に変換する文字列の言語（2 文字または 3 文字の小文字、ISO 639 言語名コード）。デフォルトでは現在の QGIS ユーザーのロケールを使用します
例	• `to_datetime('2012-05-04 12:50:00')` → 2012-05-04T12:50:00 • `to_datetime('June 29, 2019 @ 12:34','MMMM d, yyyy @ HH:mm')` → 2019-06-29T12:34（現在のロケールが 6 番目の月名に 'June' を使用する言語の場合の結果。そうでない場合はエラーが発生します） • `to_datetime('29 juin, 2019 @ 12:34','d MMMM, yyyy @ HH:mm','fr')` → 2019-06-29T12:34

to_decimal

度分秒の座標を十進数に変換します。

構文	to_decimal(value)
引数	• **value** - 度分秒の文字列
例	• `to_decimal('6° 21\'16.445')` → 6.3545680555

to_dm

座標を度、分単位に変換します。

構文	to_dm(coordinate, axis, precision, [formatting=]) 記号 [] は、オプションの引数を意味します。
引数	• **coordinate** - 緯度または経度の値 • **axis** - 座標軸。 'x' または 'y' • **precision** - 小数点以下の桁数 • **formatting** - フォーマット型を指定する。NULL（デフォルト値）、 'aligned' または 'suffix'
例	• `to_dm(6.1545681, 'x', 3)` → 6° 9.274′ • `to_dm(6.1545681, 'y', 4, 'aligned')` → 6° 09.2741′ N • `to_dm(6.1545681, 'y', 4, 'suffix')` → 6° 9.2741′ N

to_dms

座標を度、分、秒形式に変換します。

構文	to_dms(coordinate, axis, precision, [formatting=]) 記号 [] は、オプションの引数を意味します。
引数	• **coordinate** - 緯度または経度の値 • **axis** - 座標軸。 'x' または 'y' • **precision** - 小数点以下の桁数 • **formatting** - フォーマット型を指定する。NULL（デフォルト値）、 'aligned' または 'suffix'
例	• `to_dms(6.1545681, 'x', 3)` → 6 ° 9 ′ 16.445 ″ • `to_dms(6.1545681, 'y', 4, 'aligned')` → 6 ° 09 ′ 16.4452 ″ N • `to_dms(6.1545681, 'y', 4, 'suffix')` → 6 ° 9 ′ 16.4452 ″ N

to_int

文字列を整数値に変換します。整数値に変換できない場合は何も返されません（例えば '123asd' は無効な文字列です）。

構文	to_int(string)
引数	• **string** - 整数値に変換する文字列
例	• `to_int('123')` → 123

to_interval

文字列をインターバル型に変換します。日付に日、時間、月などを加減することに利用できます。

構文	to_interval(string)
引数	• **string** - インターバルを表す文字列。有効なフォーマットには、 {n} days 、 {n} hours 、 {n} months などが含まれます。
例	• `to_interval('1 day 2 hours')` → 間隔: 1.08333 日 • `to_interval( '0.5 hours' )` → 間隔: 30 分 • `to_datetime('2012-05-05 12:00:00') - to_interval('1 day 2 hours')` → 2012-05-04T10:00:00

to_real

文字列を実数値に変換します。値が実数に変換できない場合は何も返されません（例えば '123.56asd' は無効な文字列です）。フィールドの精度が変換結果の精度より小さい場合には、保存時に数値が丸められます。

構文	to_real(string)
引数	• **string** - 実数値に変換する文字列
例	• `to_real('123.45')` → 123.45

to_string

数値を文字列に変換します。

構文	to_string(number)
引数	• **number** - 整数または実数値。文字列に変換する数値
例	• `to_string(123)` → '123'

to_time

文字列を時間型オブジェクトに変換します。オプションとして、文字列をパースするためのフォーマット文字列を指定できます。フォーマットに関するドキュメントは、 QTime::fromString を参照してください。

構文	to_time(string, [format], [language]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 時刻を表す文字列 • **format** - 文字列を時間型に変換するために使用するフォーマット • **language** - 時間型に変換する文字列の言語（2 文字または 3 文字の小文字、 ISO 639 言語名コード）
例	• `to_time('12:30:01')` → 12:30:01 • `to_time('12:34','HH:mm')` → 12:34:00 • `to_time('12:34','HH:mm','fr')` → 12:34:00

13.2.6 カスタム関数

このグループには、ユーザーが作成した関数が含まれています。詳細については **関数エディタ** を参照してください。

13.2.7 日付と時刻の関数

このグループには、日付や時刻のデータを扱う関数が含まれています。このグループは、**変換関数**（to_date, to_time, to_datetime, to_interval）や **文字列関数**（format_date）グループといくつかの関数を共有しています。

注釈: フィールドに日付、日付時刻、インターバルを格納する

日付 、 **時間** 、 **日付時刻** の値をフィールドに直接格納できるかは、データソースプロバイダに依存しています（例えば、シェープファイルは **日付** の形式は可能ですが、 **日付時刻** や **時間** の形式は不可です）。以下は、このような制限がある場合に制限を克服するための案です：

- **日付** 、 **日付時刻** 、 **時間** の値は *format_date()* 関数を使用して変換し、テキスト型のフィールドに格納できます。
- **インターバル** は、日付抽出関数のどれか（例：日単位で表現されたインターバルを取得するには *day()* ）を使用した後、整数型または実数型のフィールドに格納できます。

---

age

2 つの日付または日付時刻の差を返します。

差は インターバル型 として返されるため、有用な情報を抽出するためには以下のいずれかの関数と一緒に使用する必要があります。

- `year`
- `month`
- `week`
- `day`
- `hour`
- `minute`
- `second`

構文	age(datetime1, datetime2)
引数	• **datetime1** - 時間的に後の日付を表す文字列、日付型または日付時刻型 • **datetime2** - 時間的に前の日付を表す文字列、日付型または日付時刻型
例	• `day(age('2012-05-12','2012-05-02'))` → 10 • `hour(age('2012-05-12','2012-05-02'))` → 240

datetime_from_epoch

協定世界時（Qt.UTC） 1970-01-01T00:00:00.000 から msecs ミリ秒数経過した日付時刻を Qt.LocalTime に変換した日付時刻型を返します。

構文	datetime_from_epoch(int)
引数	• **int** - 数値（ミリ秒）
例	• `datetime_from_epoch(1483225200000)` → 2017-01-01T00:00:00

day

日付型から日を取り出します。インターバル型からは日数を取り出します。

日付型バージョン

日付型または日付時刻型から、日を取り出します。

構文	day(date)
引数	• **date** - 日付型または日付時刻型の値
例	• `day('2012-05-12')` → 12

インターバル型バージョン

インターバル型の長さを日単位で計算します。

構文	day(interval)
引数	• **interval** - 日数を取得するインターバルの値
例	• `day(to_interval('3 days'))` → 3 • `day(to_interval('3 weeks 2 days'))` → 23 • `day(age('2012-01-01','2010-01-01'))` → 730

day_of_week

指定した日付または日付時刻の曜日を返します。返される値は 0 から 6 の範囲で、0 が日曜日、6 が土曜日に対応します。

構文	day_of_week(date)
引数	• **date** - 日付型または日付時刻型の値
例	• `day_of_week(to_date('2015-09-21'))` → 1

epoch

Unix エポックと指定された日付値の間の間隔をミリ秒で返します。

構文	epoch(date)
引数	• **date** - 日付型または日付時刻型の値
例	• `epoch(to_date('2017-01-01'))` → 1483203600000

format_date

日付型または文字列をカスタム書式文字列でフォーマットします。Qt の date/time フォーマット文字列を使用します。詳細は QDateTime::toString を参照してください。

構文	format_date(datetime, format, [language]) 記号 [] は、オプションの引数を意味します。
引数	• **datetime** - 日付型、時間型または日付時刻型の値 • **format** - 文字列の書式設定に使用する文字列テンプレート

式	出力
d	十の位のゼロがない日付（1 から 31）
dd	十の位にゼロのある日付（01 から 31）
ddd	短縮形のローカル曜日名（例：' 月' から ' 日'）
dddd	長いローカル曜日名（例：' 月曜日' から ' 日曜日'）
M	十の位のゼロがない月（1 から 12）
MM	十の位にゼロのある月（01 から 12）
MMM	短縮形のローカル月名（例：'1 月' から '12 月'）
MMMM	長いローカル月名（例：'1 月' から '12 月'）
yy	2 桁の年（00 から 99）
yyyy	4 桁の年

以下の式は、書式文字列の時間部分に使用します。

式	出力
h	十の位のゼロなしの時（0 から 23、AM/PM 表示の場合は 1 から 12）
hh	十の位のゼロありの時（00 から 23、AM/PM 表示の場合は 01 から 12）
H	十の位のゼロなしの時（0 から 23、AM/PM 表示の場合も同様）
HH	十の位のゼロありの時（00 から 23、AM/PM 表示の場合も同様）
m	十の位のゼロなしの分（0 から 59）
mm	十の位のゼロありの分（00 から 59）
s	十の位のゼロなしの秒（0 から 59）
ss	十の位のゼロありの秒（00 から 59）
z	数字の後に付くゼロなしのミリ秒（0 から 999）
zzz	数字の後に付くゼロありのミリ秒（000 から 999）
AP or A	AM/PM 時間として解釈します。 *AP* は 'AM' または 'PM' のいずれかです。
ap or a	AM/PM 時間として解釈します。 *ap* は 'am' または 'pm' のいずれかです。

• **language** - 日付をカスタム文字列にフォーマットする際に使用する言語（2 文字または 3 文字の小文字、ISO 639 言語名コード ）。デフォルトでは現在の QGIS ユーザーのロケールを使用します

例

• `format_date('2012-05-15','dd.MM.yyyy')` → '15.05.2012'
• `format_date('2012-05-15','d MMMM yyyy','fr')` → '15 mai 2012'
• `format_date('2012-05-15','dddd')` → 'Tuesday'（現在のロケールが English 版の場合の結果）
• `format_date('2012-05-15 13:54:20','dd.MM.yy')` → '15.05.12'
• `format_date('13:54:20','hh:mm AP')` → '01:54 午後'

hour

時間型または日付時刻型から時の部分を取り出します。インターバル型からは時間数を取り出します。

時間型バージョン

時間型または日付時刻型から時の部分を取り出します。

構文	hour(datetime)
引数	• **datetime** - 時間型または日付時刻型の値
例	• `hour( to_datetime('2012-07-22 13:24:57') )` → 13

インターバル型バージョン

インターバル型の長さを時間単位で計算します。

構文	hour(interval)
引数	• **interval** - 時間数を取得するインターバルの値
例	• `hour(to_interval('3 hours'))` → 3 • `hour(age('2012-07-22T13:00:00','2012-07-22T10:00:00'))` → 3 • `hour(age('2012-01-01','2010-01-01'))` → 17520

make_date

年、月、日の数字から日付型の値を作ります。

構文	make_date(year, month, day)
引数	• **year** - 年の数字。1 から 99 はそのままの値として解釈される。0 年は無効な値 • **month** - 月の数字 • **day** - 日の数字。1 から始まる
例	• `make_date(2020,5,4)` → date value 2020-05-04

make_datetime

年、月、日、時、分、秒の数字から日付時刻型の値を作ります。

構文	make_datetime(year, month, day, hour, minute, second)
引数	• **year** - 年の数字。1 から 99 はそのままの値として解釈される。0 年は無効な値 • **month** - 月の数字 • **day** - 日の数字。1 から始まる • **hour** - 時間の数字 • **minute** - 分の数字 • **second** - 秒の数字（小数点以下の数字はミリ秒）
例	• `make_datetime(2020,5,4,13,45,30.5)` → datetime value 2020-05-04 13:45:30.500

make_interval

年、月、日、時、分、秒の数字からインターバル型の値を作ります。

構文	make_interval([years=0], [months=0], [weeks=0], [days=0], [hours=0], [minutes=0], [seconds=0]) 記号 [] は、オプションの引数を意味します。
引数	• **years** - 年の数（1 年の長さを 365.25 日とする） • **months** - 月の数（1 月の長さを 30 日とする） • **weeks** - 週の数 • **days** - 日の数 • **hours** - 時間の数 • **minutes** - 分の数 • **seconds** - 秒の数
例	• `make_interval(hours:=3)` → 間隔: 3 時間 • `make_interval(days:=2, hours:=3)` → 間隔: 2.125 日 • `make_interval(minutes:=0.5, seconds:=5)` → 間隔: 35 秒

make_time

時、分、秒の数字から時間型の値を作ります。

構文	make_time(hour, minute, second)
引数	• **hour** - 時間の数字 • **minute** - 分の数字 • **second** - 秒の数字（小数点以下の数字はミリ秒）
例	• `make_time(13,45,30.5)` → time value 13:45:30.500

minute

日付時刻型または時間型から分の部分を取り出します。インターバル型からは分数を取り出します。

時間型バージョン

時間型または日付時刻型から分の部分を取り出します。

構文	minute(datetime)
引数	• **datetime** - 時間型または日付時刻型の値
例	• `minute( to_datetime('2012-07-22 13:24:57') )` → 24

インターバル型バージョン

インターバル型の長さを分単位で計算します。

構文	minute(interval)
引数	• **interval** - 分数を取得するインターバルの値
例	• `minute(to_interval('3 minutes'))` → 3 • `minute(age('2012-07-22T00:20:00','2012-07-22T00:00:00'))` → 20 • `minute(age('2012-01-01','2010-01-01'))` → 1051200

month

日付型から月を取り出します。インターバル型からは月数を取り出します。

日付型バージョン

日付型または日付時刻型から月の部分を取り出します。

構文	month(date)
引数	• **date** - 日付型または日付時刻型の値
例	• `month('2012-05-12')` → 05

インターバル型バージョン

インターバル型の長さを月単位で計算します。

構文	month(interval)
引数	• **interval** - 月数を取得するインターバルの値
例	• `month(to_interval('3 months'))` → 3 • `month(age('2012-01-01','2010-01-01'))` → 4.03333

now

現在の日付と時刻を返します。この関数は静的で、評価の最中は一貫した結果を返します。返される時刻は、この式が準備されたときの時刻です。

構文	now()
例	• `now()` → 2012-07-22T13:24:57

second

時間型または日付時刻型から秒の部分を取り出します。インターバル型からは秒数を取り出します。

時間型バージョン

時間型または日付時刻型から秒の部分を取り出します。

構文	second(datetime)
引数	• **datetime** - 時間型または日付時刻型の値
例	• `second( to_datetime('2012-07-22 13:24:57') )` → 57

インターバル型バージョン

インターバル型の長さを秒単位で計算します。

構文	second(interval)
引数	• **interval** - 秒数を取得するインターバルの値
例	• `second(to_interval('3 minutes'))` → 180 • `second(age('2012-07-22T00:20:00','2012-07-22T00:00:00'))` → 1200 • `second(age('2012-01-01','2010-01-01'))` → 63072000

to_date

文字列を日付型オブジェクトに変換します。オプションとして、文字列をパースするためのフォーマット文字列を指定できます。フォーマットに関するドキュメントは、 QDate::fromString または format_date 関数のドキュメントを参照してください。デフォルトでは現在の QGIS ユーザーのロケールを使用します。

構文	to_date(string, [format], [language]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 日付を表す文字列 • **format** - 文字列を日付型に変換するために使用するフォーマット • **language** - 日付型に変換する文字列の言語（2 文字または 3 文字の小文字、ISO 639 言語名コード）。デフォルトでは現在の QGIS ユーザーのロケールを使用します
例	• `to_date('2012-05-04')` → 2012-05-04 • `to_date('June 29, 2019','MMMM d, yyyy')` → 2019-06-29 （現在のロケールが 6 番目の月名に 'June' を使用する言語の場合の結果。そうでない場合はエラーが発生します） • `to_date('29 juin, 2019','d MMMM, yyyy','fr')` → 2019-06-29

to_datetime

文字列を日付時刻型オブジェクトに変換します。オプションとして、文字列をパースするためのフォーマット文字列を指定できます。フォーマットに関するドキュメントは、QDate::fromString 、QTime::fromString または format_date 関数のドキュメントを参照してください。デフォルトでは現在の QGIS ユーザーのロケールを使用します。

構文	to_datetime(string, [format], [language]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 日付時刻を表す文字列 • **format** - 文字列を日付時刻型に変換するために使用するフォーマット • **language** - 日付時刻型に変換する文字列の言語（2 文字または 3 文字の小文字、ISO 639 言語名コード）。デフォルトでは現在の QGIS ユーザーのロケールを使用します
例	• `to_datetime('2012-05-04 12:50:00')` → 2012-05-04T12:50:00 • `to_datetime('June 29, 2019 @ 12:34','MMMM d, yyyy @ HH:mm')` → 2019-06-29T12:34（現在のロケールが 6 番目の月名に 'June' を使用する言語の場合の結果。そうでない場合はエラーが発生します） • `to_datetime('29 juin, 2019 @ 12:34','d MMMM, yyyy @ HH:mm','fr')` → 2019-06-29T12:34

to_interval

文字列をインターバル型に変換します。日付に日、時間、月などを加減することに利用できます。

構文	to_interval(string)
引数	• **string** - インターバルを表す文字列。有効なフォーマットには、 {n} days 、 {n} hours 、 {n} months などが含まれます。
例	• `to_interval('1 day 2 hours')` → 間隔: 1.08333 日 • `to_interval( '0.5 hours' )` → 間隔: 30 分 • `to_datetime('2012-05-05 12:00:00') - to_interval('1 day 2 hours')` → 2012-05-04T10:00:00

to_time

文字列を時間型オブジェクトに変換します。オプションとして、文字列をパースするためのフォーマット文字列を指定できます。フォーマットに関するドキュメントは、 QTime::fromString を参照してください。

構文	to_time(string, [format], [language]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 時刻を表す文字列 • **format** - 文字列を時間型に変換するために使用するフォーマット • **language** - 時間型に変換する文字列の言語（2 文字または 3 文字の小文字、 ISO 639 言語名コード）
例	• `to_time('12:30:01')` → 12:30:01 • `to_time('12:34','HH:mm')` → 12:34:00 • `to_time('12:34','HH:mm','fr')` → 12:34:00

week

日付型から週番号を取り出します。インターバル型からは週単位の長さを取り出します。

日付型バージョン

日付型または日付時刻型から週番号を取り出します。

構文	week(date)
引数	• **date** - 日付型または日付時刻型の値
例	• `week('2012-05-12')` → 19

インターバル型バージョン

インターバル型の長さを週単位で計算します。

構文	week(interval)
引数	• **interval** - 月数を取得するインターバルの値
例	• `week(to_interval('3 weeks'))` → 3 • `week(age('2012-01-01','2010-01-01'))` → 104.285

year

日付型から年を取り出します。インターバル型からは年数を取り出します。

日付型バージョン

日付型または日付時刻型から年の部分を取り出します。

構文	year(date)
引数	• **date** - 日付型または日付時刻型の値
例	• `year('2012-05-12')` → 2012

インターバル型バージョン

インターバル型の長さを年単位で計算します。

構文	year(interval)
引数	• **interval** - 年数を取得するインターバルの値
例	• `year(to_interval('3 years'))` → 3 • `year(age('2012-01-01','2010-01-01'))` → 1.9986

例

これらの関数に加えて、 - （マイナス）演算子を使用して日付型や日付時刻型または時刻型どうしの減算を行うと、インターバル型のデータを返します。

+（プラス）演算子や -（マイナス）演算子を使用して、日付型や日付時刻型、時間型のデータにインターバル型のデータを加算または減算すると、日時時刻型のデータを返します。

- QGIS 3.0 リリースまでの日数を取得します。

```
to_date('2017-09-29') - to_date(now())
-- Returns <interval: 203 days>
```

- リリース時刻も含めて計算します：

```
to_datetime('2017-09-29 12:00:00') - now()
-- Returns <interval: 202.49 days>
```

- 今から 100 日の日時を取得します。

```
now() + to_interval('100 days')
-- Returns <datetime: 2017-06-18 01:00:00>
```

13.2.8 フィールドと値

アクティブレイヤのフィールドのリストと、特別な値を格納します。フィールドのリストには、データセットに格納されているもの、 *virtual* と *auxiliary* のもの、 *joins* のものが含まれます。

フィールド名をダブルクリックすると、そのフィールドが式に追加されます。式への追加は、フィールド名（ダブルクォート囲みを推奨）またはその **別名** を直接入力することでも可能です。

式で使用したいフィールドの値を取得するには、適切なフィールドを選択し、表示されたウィジェットで *10* **個のサンプル** または **全ユニーク** のどちらかのボタンを押します。すると、フィールドの値が表示され、リストの上部にある **検索** ボックスを使用して結果をフィルタリングできます。サンプル値取得は、フィールド名の上で右クリックしてもアクセスできます。

式に値を追加するには、フィールド値リスト内の値をダブルクリックします。値が文字列型の場合はシングルクォートで囲まれますが、それ以外はクォートで囲みません。

NULL

NULL に等しい値です。

構文	NULL
例	• `NULL` → NULL 値

注釈: NULL 値かどうかをテストする場合には、 *IS NULL* 式または *IS NOT NULL* 式を使用してください。

13.2.9 ファイルとパスの関数

このグループには、ファイルとパス名を操作する関数が含まれます。

base_file_name

ディレクトリや拡張子を取り除いた、ファイルのベース名を返します。

構文	base_file_name(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `base_file_name('/home/qgis/data/country_boundaries.shp')` → 'country_boundaries'

exif

画像ファイルから exif タグの値を取得します。

構文	exif(path, [tag]) 記号 [] は、オプションの引数を意味します。
引数	• **path** - 画像ファイルのパスまたはマップレイヤの値。マップレイヤの値が指定された場合、そのレイヤーのファイルソースが使用されます。 • **tag** - 取得したいタグ。指定しない場合には、全ての exif タグの値がマップ型で返ります
例	• `exif('/my/photo.jpg','Exif.Image.Orientation')` → 0

file_exists

ファイルパスが存在する場合は TRUE を返します。

構文	file_exists(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `file_exists('/home/qgis/data/country_boundaries.shp')` → TRUE

file_name

（拡張子付きの）ファイル名を返します。ディレクトリは含みません。

構文	file_name(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `file_name('/home/qgis/data/country_boundaries.shp')` → 'country_boundaries.shp'

file_path

ファイルパスのディレクトリ部分を返します。戻り値にはファイル名は含まれません。

構文	file_path(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `file_path('/home/qgis/data/country_boundaries.shp')` → '/home/qgis/data'

file_size

ファイルのサイズを（バイト単位で）返します。

構文	file_size(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `file_size('/home/qgis/data/country_boundaries.geojson')` → 5674

file_suffix

ファイルパスから拡張子（末尾部分）を返します。

構文	file_suffix(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `file_suffix('/home/qgis/data/country_boundaries.shp')` → 'shp'

is_directory

パスがディレクトリに対応する場合、TRUE を返します。

構文	is_directory(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `is_directory('/home/qgis/data/country_boundaries.shp')` → FALSE • `is_directory('/home/qgis/data/')` → TRUE

is_file

パスがファイルに対応する場合、TRUE を返します。

構文	is_file(path)
引数	• **path** - ファイルパス又はマップレイヤの値。マップレイヤの値が指定された場合は、そのレイヤのファイル元が使われる。
例	• `is_file('/home/qgis/data/country_boundaries.shp')` → TRUE • `is_file('/home/qgis/data/')` → FALSE

13.2.10 フォーム関数

このグループには、属性フォームに関連した式ダイアログのみで利用できる関数が含まれます。例えば、**フィールドのウィジェット** の設定などで利用できます。

current_parent_value

この関数は埋め込みフォームでのみ使用可能で、現在編集中の親フォームの未保存の属性値を返します。これは、親レイヤで編集中または親レイヤに追加前の親地物の実際の値とは異なります。「値のリレーション」ウィジェットのフィルタ式で使用する場合には、フォームが埋め込みフォーム状態で使われない場合にもレイヤから実際の親地物の値を取得できるよう、 'coalesce()' 関数でラップする必要があります。

構文	current_parent_value(field_name)
引数	• **field_name** - 現在の親フォームのフィールド名
例	• `current_parent_value( 'FIELD_NAME' )` → 親フォームの 'FIELD_NAME' フィールドの現在の値

current_value

現在編集中のフォームまたはテーブル行で、現在のフィールドの未保存の値を返します。これは、現在編集中の地物またはまだレイヤに追加されていない地物の実際の属性値とは異なります。

構文	current_value(field_name)
引数	• **field_name** - 現在のフォームまたはテーブル行のフィールド名
例	• `current_value( 'FIELD_NAME' )` → 'FIELD_NAME' フィールドの現在の値

13.2.11 ファジー・マッチング関数

このグループには、値間でファジィ比較をするための関数が含まれています。

hamming_distance

2 つの文字列間のハミング距離を返します。この距離は、入力文字列の対応する位置にある異なった文字の数です。入力文字列は同じ長さでなければならず、大文字と小文字を区別して比較されます。

構文	hamming_distance(string1, string2)
引数	• **string1** - 文字列 • **string2** - 文字列
例	• `hamming_distance('abc','xec')` → 2 • `hamming_distance('abc','ABc')` → 2 • `hamming_distance(upper('abc'),upper('ABC'))` → 0 • `hamming_distance('abc','abcd')` → NULL

levenshtein

2 つの文字列間のレーベンシュタイン編集距離を返します。これは、ある文字列を別の文字列に変更するために必要な文字編集（挿入、削除または置換）の最小数に相当します。

レーベンシュタイン距離は、2 つの文字列間の類似性の尺度です。距離が小さいほど文字列はより似ており、距離が長いほど文字列が異なっていることを示します。距離は大文字と小文字を区別します。

構文	levenshtein(string1, string2)
引数	• **string1** - 文字列 • **string2** - 文字列
例	• `levenshtein('kittens','mitten')` → 2 • `levenshtein('Kitten','kitten')` → 1 • `levenshtein(upper('Kitten'),upper('kitten'))` → 0

longest_common_substring

２つの文字列間の最も長い共通部分を返します。例えば、 "ABABC" と "BABCA" の最も長い共通部分は "BABC" です。部分文字列は大文字と小文字を区別します。

構文	longest_common_substring(string1, string2)
引数	• **string1** - 文字列 • **string2** - 文字列
例	• `longest_common_substring('ABABC','BABCA')` → 'BABC' • `longest_common_substring('abcDeF','abcdef')` → 'abc' • `longest_common_substring(upper('abcDeF'),upper('abcdex'))` → 'ABCDE'

soundex

文字列の Soundex 表現を返します。Soundex は音声マッチングアルゴリズムで、類似した音の文字列は同じ Soundex コードで表現されます。

構文	soundex(string)
引数	• **string** - 文字列
例	• `soundex('robert')` → 'R163' • `soundex('rupert')` → 'R163' • `soundex('rubin')` → 'R150'

13.2.12 一般関数

このグループには、さまざまな汎用関数が含まれます。

env

環境変数を取得し、その内容を文字列として返します。変数が見つからない場合、NULL を返します。この関数は、ドライブ文字やパスプレフィックスなどのシステム固有の設定を取得するために有用です。環境変数の定義はオペレーティングシステムによって異なりますので、システム管理者に確認するかオペレーティングシステムのマニュアルを参照して、環境変数の設定方法を確認してください。

構文	env(name)
引数	• **name** - 値を取得したい環境変数の名前
例	• `env( 'LANG' )` → 'en_US.UTF-8' • `env( 'MY_OWN_PREFIX_VAR' )` → 'Z:' • `env( 'I_DO_NOT_EXIST' )` → NULL

eval

文字列で渡された式を評価します。これを使えば、コンテキスト変数またはフィールドとして渡されたパラメータを動的に展開できます。

構文	eval(expression)
引数	• **expression** - 式を表す文字列
例	• `eval('\'nice\'')` → 'nice' • `eval(@expression_var)` → [@expression_var を評価した結果]

eval_template

文字列で渡されたテンプレートを評価します。これを使えば、コンテキスト変数またはフィールドとして渡されたパラメータを動的に展開できます。

構文	eval_template(template)
引数	• **template** - テンプレート文字列
例	• `eval_template('QGIS [% upper(\'rocks\') %]')` → QGIS ROCKS

is_layer_visible

指定されたレイヤが表示されているなら TRUE を返します。

構文	is_layer_visible(layer)
引数	• **layer** - レイヤ名かレイヤ ID のどちらかを表す文字列
例	• `is_layer_visible('baseraster')` → TRUE

mime_type

バイナリデータの MIME タイプを返します。

構文	mime_type(bytes)
引数	• **bytes** - バイナリデータ
例	• `mime_type('<html><body></body></html>')` → text/html • `mime_type(from_base64('R0lGODlhAQABAAAAACH5BAEKAAEALAAAAAABAAEAAAIAOw=='))` → image/gif

var

指定された変数に格納された値を返します。

構文	var(name)
引数	• **name** - 変数名
例	• `var('qgis_version')` → '2.12'

参考：デフォルトの **変数** のリスト

with_variable

この関数は、第 3 引数の式で使われる変数を設定します。同じ計算結果を別の場所で繰り返し使用する必要があるような、複雑な式を構成する際に使われます。

構文	with_variable(name, value, expression)
引数	• **name** - 設定する変数の名前 • **value** - 設定する値 • **expression** - 変数が使用できる式
例	• `with_variable('my_sum', 1 + 2 + 3, @my_sum * 2 + @my_sum * 5)` → 42

13.2.13 ジオメトリ関数

このグループは、ジオメトリオブジェクトを操作する関数（例えば buffer、 transform、 $area など）が含まれます。

affine_transform

アフィン変換後のジオメトリを返します。計算はこのジオメトリの空間参照系で行われます。変換操作は、スケーリング、回転、平行移動の順で行われます。Z 座標や M 値のオフセットが指定されているものの、ジオメトリの座標が Z 座標や M 値を持たない場合には、これらがジオメトリ座標に追加されます。

構文	affine_transform(geometry, delta_x, delta_y, rotation_z, scale_x, scale_y, [delta_z=0], [delta_m=0], [scale_z=1], [scale_m=1]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **delta_x** - x 軸方向移動量 • **delta_y** - y 軸方向移動量 • **rotation_z** - z 軸まわりの回転量（度単位、反時計回り） • **scale_x** - x 軸方向スケーリング係数 • **scale_y** - y 軸方向スケーリング係数 • **delta_z** - z 軸方向移動量 • **delta_m** - m 値の増分 • **scale_z** - z 軸方向スケーリング係数 • **scale_m** - m 値のスケーリング係数
例	• `geom_to_wkt(affine_transform(geom_from_wkt('LINESTRING(1 1, 2 2)'), 2, 2, 0, 1, 1))` → 'LineString (3 3, 4 4)' • `geom_to_wkt(affine_transform(geom_from_wkt('POLYGON((0 0, 0 3, 2 2, 0 0))'), 0, 0, -90, 1, 2))` → 'Polygon ((0 0, 6 0, 4 -2, 0 0))' • `geom_to_wkt(affine_transform(geom_from_wkt('POINT(3 1)'), 0, 0, 0, 1, 1, 5, 0))` → 'PointZ (3 1 5)'

angle_at_vertex

ラインストリングジオメトリ上の指定された頂点について、ジオメトリの二分角（平均角度）を返します。角度は度単位で、北から時計回りで測ります。

構文	angle_at_vertex(geometry, vertex)
引数	• **geometry** - ラインストリングジオメトリ • **vertex** - 頂点のインデックスで、最初の点は 0 から数える。負の値の場合には、全体の頂点数からその絶対値を引いたインデックスの頂点が選択される
例	• `angle_at_vertex(geometry:=geom_from_wkt('LineString(0 0, 10 0, 10 10)'),vertex:=1)` → 45.0

apply_dash_pattern

ジオメトリにダッシュパターンを適用し、入力ジオメトリを指定されたパターンで各線/リングに沿ってストロークさせた MultiLineString ジオメトリを返します。

構文	apply_dash_pattern(geometry, pattern, [start_rule=no_rule], [end_rule=no_rule], [adjustment=both], [pattern_offset=0]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ（ (multi)linestrings 又は (multi)polygons を受け付けます）。 • **pattern** - ダッシュパターン。ダッシュとギャップの長さを表す数値の配列です。偶数個の要素を含む必要があります。 • **start_rule** - パターンの開始を制約するためのオプションのルールです。有効な値は 'no_rule', 'full_dash', 'half_dash', 'full_gap', 'half_gap' です。 • **end_rule** - パターンの終わりを制約するためのオプションのルールです。有効な値は 'no_rule', 'full_dash', 'half_dash', 'full_gap', 'half_gap' です。 • **adjustment** - 求めるパターンルールに合わせるために、パターンのどの部分を調整するかを指定するオプションのルールです。有効な値は 'both'、'dash'、'gap' です。 • **pattern_offset** - パターンに沿った特定の距離で開始することを指定するオプションの距離です。
例	• `geom_to_wkt(apply_dash_pattern(geom_from_wkt('LINESTRING(1 1, 10 1)'), array(3, 1)))` → MultiLineString ((1 1, 4 1),(5 1, 8 1),(9 1, 10 1, 10 1)) • `geom_to_wkt(apply_dash_pattern(geom_from_wkt('LINESTRING(1 1, 10 1)'), array(3, 1), start_rule:='half_dash'))` → MultiLineString ((1 1, 2.5 1),(3.5 1, 6.5 1),(7.5 1, 10 1, 10 1))

$area

地物の面積を返します。この関数で計算される面積には、現在のプロジェクトの楕円体設定と面積単位設定の両方が反映されます。例えば、プロジェクトに回転楕円体が設定されている場合、楕円体面積になり、設定されていない場合、平面上の面積になります。

構文	$area
例	• `$area` → 42

area

ポリゴンオブジェクトのジオメトリの面積を返します。面積はこのジオメトリの空間参照系（SRS）で常に平面的に計算され、面積の単位は空間参照系の単位と一致します。この面積は、プロジェクトの楕円体と面積単位設定にもとづいた楕円体計算が行われる $area 関数による面積とは異なります。

構文	area(geometry)
引数	• **geometry** - ポリゴンジオメトリオブジェクト
例	• `area(geom_from_wkt('POLYGON((0 0, 4 0, 4 2, 0 2, 0 0))'))` → 8.0

azimuth

北を基準して時計回りに測定された point_a から point_b への方位角をラジアン単位で返します。

構文	azimuth(point_a, point_b)
引数	• **point_a** - ポイントジオメトリ • **point_b** - ポイントジオメトリ
例	• `degrees( azimuth( make_point(25, 45), make_point(75, 100) ) )` → 42.273689 • `degrees( azimuth( make_point(75, 100), make_point(25,45) ) )` → 222.273689

boundary

ジオメトリの組み合わせ的境界のクロージャ（すなわちジオメトリの位相的境界）を返します。例えば、ポリゴンジオメトリには、ポリゴンの各リングのラインストリングからなる境界があります。ポイントやジオメトリコレクションのように、定義された境界を持たないジオメトリについては、NULL が返されます。

構文	boundary(geometry)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkt(boundary(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')))` → 'LineString(1 1,0 0,-1 1,1 1)' • `geom_to_wkt(boundary(geom_from_wkt('LineString(1 1,0 0,-1 1)')))` → 'MultiPoint ((1 1),(-1 1))'

参考： **トポロジカル境界** アルゴリズム

bounds

入力ジオメトリのバウンディングボックスを表すジオメトリを返します。計算はこのジオメトリの空間参照系で行われます。

構文	bounds(geometry)
引数	• **geometry** - ジオメトリ
例	• `bounds($geometry)` → 現在の地物のジオメトリのバウンディングボックス • `geom_to_wkt(bounds(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')))` → 'Polygon ((-1 0, 1 0, 1 1, -1 1, -1 0))'

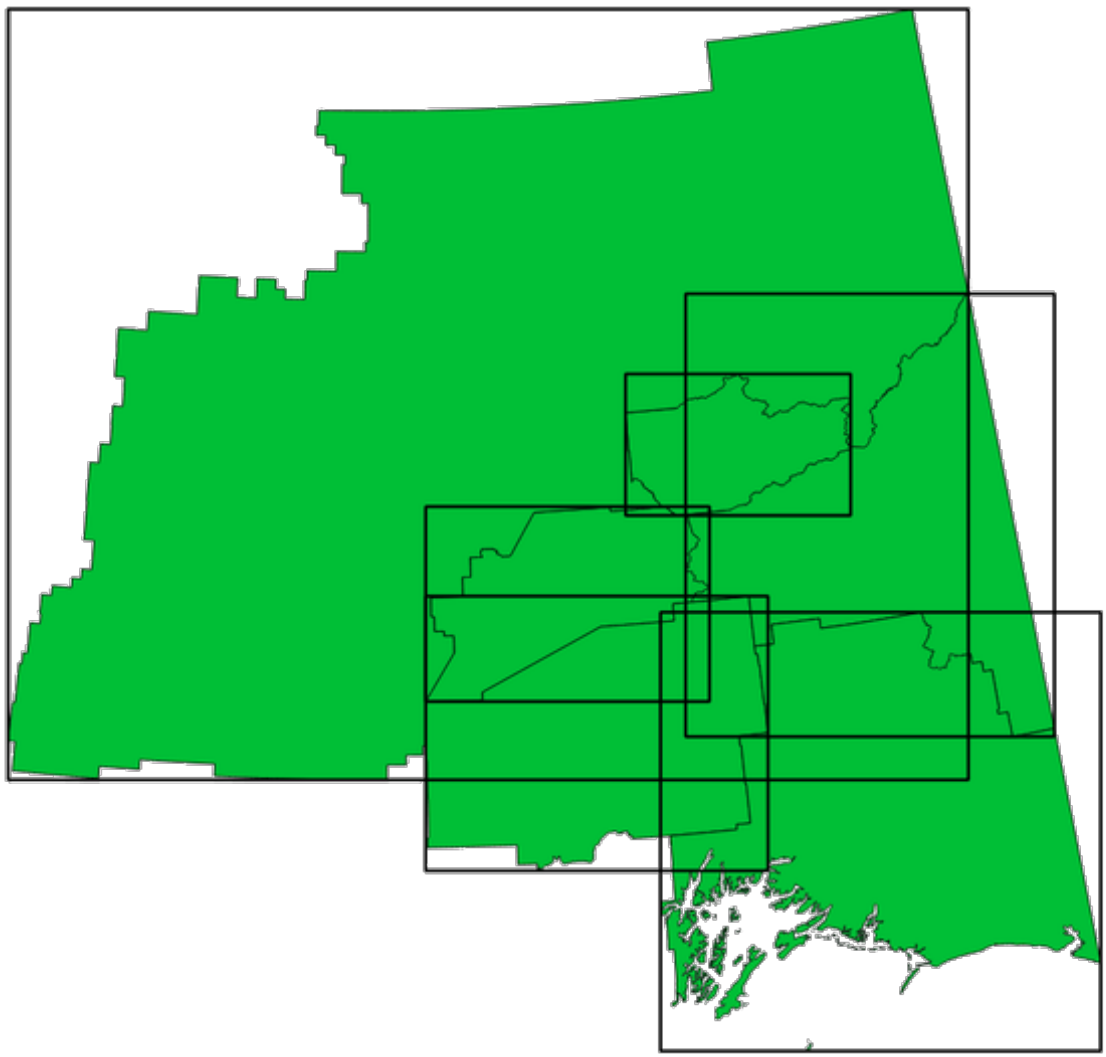


図 13.4: 黒色のラインは各ポリゴン地物のバウンディングボックスを表す

参考： *BBox* **の出力** アルゴリズム

bounds_height

ジオメトリのバウンディングボックスの高さを返します。計算はこのジオメトリの空間参照系で行われます。

構文	bounds_height(geometry)
引数	• **geometry** - ジオメトリ
例	• `bounds_height($geometry)` → 現在の地物のジオメトリのバウンディングボックスの高さ • `bounds_height(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))'))` → 1

bounds_width

ジオメトリのバウンディングボックスの幅を返します。計算はこのジオメトリの空間参照系で行われます。

構文	bounds_width(geometry)
引数	• **geometry** - ジオメトリ
例	• `bounds_width($geometry)` → 現在の地物のジオメトリのバウンディングボックスの幅 • `bounds_width(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))'))` → 2

buffer

ジオメトリからの距離がある距離以下のすべてのポイントを表すジオメトリを返します。計算はこのジオメトリの空間参照系で行なわれます。

構文	buffer(geometry, distance, [segments=8], [cap='round'], [join='round'], [miter_limit=2]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **distance** - レイヤの単位で表したバッファ距離 • **segments** - 継ぎ目スタイルに round を使用する場合に、円の四分の一を表現するのに使用するセグメントの数。数値が大きいほど、より多数のノードからなる、滑らかなバッファが生成されます。 • **cap** - バッファの終端スタイル。有効な値は 'round'、 'flat' または 'square' です • **join** - バッファの結合スタイル。有効な値は 'round'、 'bevel' または 'miter' です • **miter_limit** - miter 距離の上限値（結合スタイルに 'miter' を設定した場合に使用）
例	• `buffer($geometry, 10.5)` → 現在の地物ジオメトリから 10.5 単位でバッファリングされたポリゴン

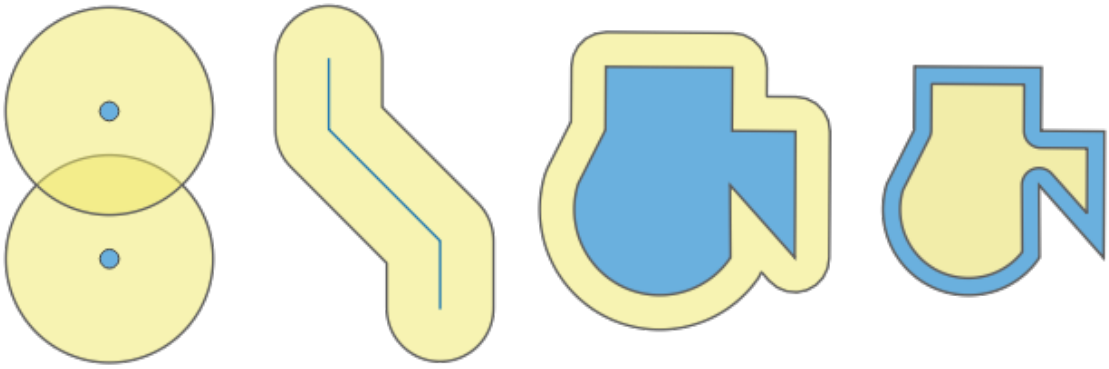


図 13.5: バッファ（黄色）。正のバッファを持ったポイント、ライン、ポリゴン及び負のバッファを持ったポリゴン

参考： **バッファ（*buffer*）** アルゴリズム

buffer_by_m

ラインジオメトリに沿ってバッファを作成します。バッファの直径は、ラインの頂点の m 値に従って変化します。

構文	buffer_by_m(geometry, [segments=8]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - 入力ジオメトリ。m 値を持った（マルチ）ラインジオメトリである必要があります。 • **segments** - バッファの四分円を近似するセグメント数
例	• `buffer_by_m(geometry:=geom_from_wkt('LINESTRINGM(1 2 0.5, 4 2 0.2)'),segments:=8)` → ラインストリングジオメトリに沿って、直径が 0.5 から 0.2 に変化する可変幅バッファ

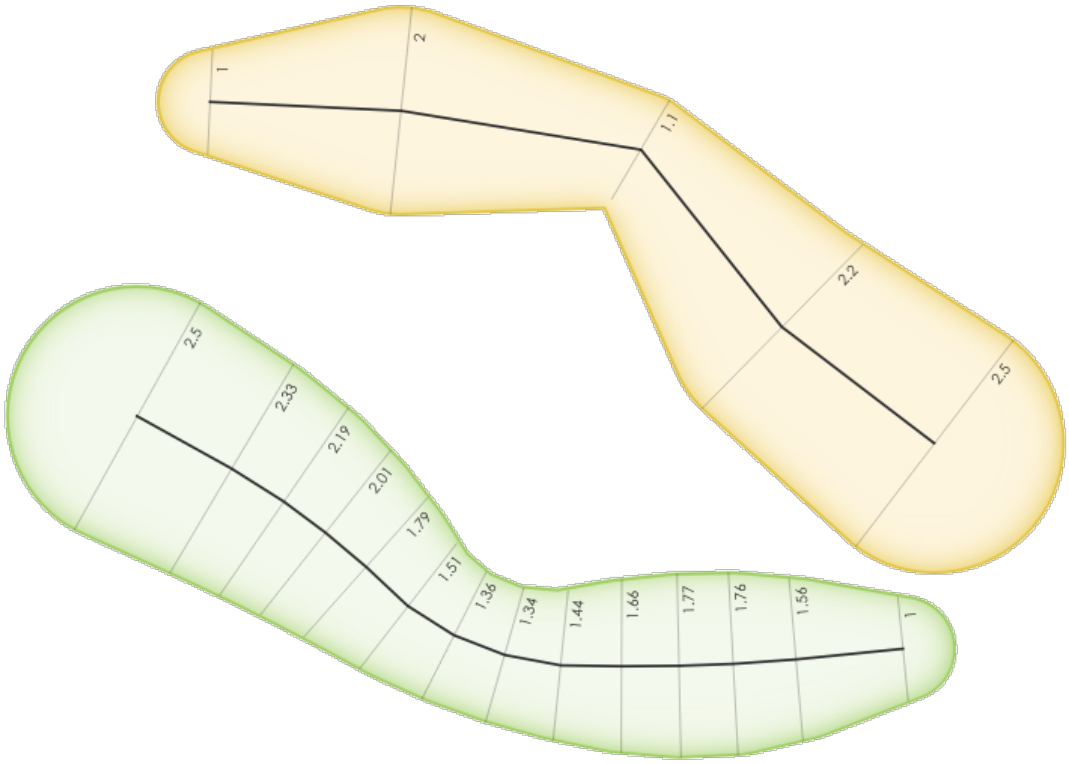


図 13.6: 頂点上の m 値を使用したライン地物のバッファ

参考： **可変幅バッファ（*M* 値使用）** アルゴリズム

centroid

ジオメトリの重心を返します。

構文	centroid(geometry)
引数	• **geometry** - ジオメトリ
例	• `centroid($geometry)` → ポイントジオメトリ

参考： **重心** アルゴリズム

close_line

入力されたラインストリングが閉じていない場合には、最初の点を行末に追加して、閉じたラインストリングを返します。ジオメトリがラインストリングまたはマルチラインストリングでない場合、結果は NULL になります。

構文	close_line(geometry)
引数	• **geometry** - ラインストリングジオメトリ
例	• `geom_to_wkt(close_line(geom_from_wkt('LINESTRING(0 0, 1 0, 1 1)')))` → 'LineString (0 0, 1 0, 1 1, 0 0)' • `geom_to_wkt(close_line(geom_from_wkt('LINESTRING(0 0, 1 0, 1 1, 0 0)')))` → 'LineString (0 0, 1 0, 1 1, 0 0)'

closest_point

geometry2 に最も近い、geometry1 上の点を返します。

構文	closest_point(geometry1, geometry2)
引数	• **geometry1** - 検索対象のジオメトリ • **geometry2** - 基準となるジオメトリ
例	• `geom_to_wkt(closest_point(geom_from_wkt('LINESTRING (20 80, 98 190, 110 180, 50 75 )'),geom_from_wkt('POINT(100 100)')))` → 'Point(73.0769 115.384)'

collect_geometries

ジオメトリを集約したマルチパートのジオメトリを返します。

引数バージョン

ジオメトリのパーツは、関数の引数として個別に指定します。

構文	collect_geometries(geometry1, geometry2, ...)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkt(collect_geometries(make_point(1,2), make_point(3,4), make_point(5,6)))` → 'MultiPoint ((1 2),(3 4),(5 6))'

配列バージョン

ジオメトリのパーツは、ジオメトリパーツの配列として指定します。

構文	collect_geometries(array)
引数	• **array** - ジオメトリオブジェクトの配列
例	• `geom_to_wkt(collect_geometries(array(make_point(1,2), make_point(3,4), make_point(5,6))))` → 'MultiPoint ((1 2),(3 4),(5 6))'

参考： **シングルパートをマルチパートに集約** アルゴリズム

combine

２つのジオメトリの結合を返します。

構文	combine(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `geom_to_wkt( combine( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 2 1)' ) ) )` → 'MULTILINESTRING((4 4, 2 1), (3 3, 4 4), (4 4, 5 5))' • `geom_to_wkt( combine( geom_from_wkt( 'LINESTRING(3 3, 4 4)' ), geom_from_wkt( 'LINESTRING(3 3, 6 6, 2 1)' ) ) )` → 'LINESTRING(3 3, 4 4, 6 6, 2 1)'

concave_hull

ジオメトリのすべてのポイントを含む、凹型のポリゴンを返します

構文	concave_hull(geometry, target_percent, [allow_holes=False]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **target_percent** - 解が凸包に近づこうとする面積の割合。target_percent が 1 の場合、凸包と同じ結果になります。target_percent を 0 から 0.99 の間に設定すると、凸包よりも面積が小さくなる結果が得られます。 • **allow_holes** - 出力ジオメトリに穴を許すかどうかを指定するオプションの引数です。デフォルトは FALSE で、出力ジオメトリに穴を含めないようにするには TRUE に設定します。
例	• `geom_to_wkt(concave_hull(geom_from_wkt('MULTILINESTRING((106 164, 30 112,74 70,82 112,130 94,130 62,122 40,156 32,162 76,172 88), (132 178,134 148,128 136,96 128,132 108,150 130,170 142,174 110, 156 96,158 90,158 88),(22 64,66 28,94 38,94 68,114 76,112 30,132 10,168 18,178 34,186 52,184 74,190 100,190 122,182 148,178 170,176 184,156 164,146 178,132 186,92 182,56 158,36 150,62 150,76 128,88 118))'), 0.99))` → 'Polygon ((30 112, 36 150, 92 182, 132 186, 176 184, 190 122, 190 100, 186 52, 178 34, 168 18, 132 10, 112 30, 66 28, 22 64, 30 112))'

参考: *convex_hull*

contains

あるジオメトリが他のものを含むかどうかを判定します。ジオメトリ 1 の外側にジオメトリ 2 の点がなく、かつジオメトリ 2 の内側の少なくとも 1 点がジオメトリ 1 の内側にある場合にのみ、TRUE を返します。

構文	contains(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `contains( geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))' ), geom_from_wkt( 'POINT(0.5 0.5 )' ) )` → TRUE • `contains( geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → FALSE

参考： *overlay_contains*

convex_hull

ジオメトリの凸包を返します。これは、セット内のすべてのジオメトリを囲む、最小の凸多角形のジオメトリです。

構文	convex_hull(geometry)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkt( convex_hull( geom_from_wkt( 'LINESTRING(3 3, 4 4, 4 10)' ) ) )` → 'POLYGON((3 3, 4 10, 4 4, 3 3))'

参考： *concave_hull*, **凸包（*convex hull*）** アルゴリズム

crosses

あるジオメトリが他のジオメトリと交差しているかどうかを判定します。与えられたジオメトリが、すべてではないが、内側の点をいくつか共有している場合、TRUE を返します。

構文	crosses(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `crosses( geom_from_wkt( 'LINESTRING(3 5, 4 4, 5 3)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `crosses( geom_from_wkt( 'POINT(4 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → FALSE

参考： *overlay_crosses*

densify_by_count

ポリゴンまたはラインレイヤのジオメトリを受け取り、元のジオメトリよりも頂点数が多いジオメトリを新たに生成します。

構文	densify_by_count(geometry, vertices)
引数	• **geometry** - ジオメトリ（ (multi)linestrings 又は (multi)polygons を受け付けます)。 • **vertices** - （セグメント毎に）追加する頂点の数
例	• `geom_to_wkt(densify_by_count(geom_from_wkt('LINESTRING(1 1, 10 1)'), 3))` → LineString (1 1, 3.25 1, 5.5 1, 7.75 1, 10 1)

参考： **頂点の高密度化（個数ベース）** アルゴリズム

densify_by_distance

ポリゴン又はラインレイヤのジオメトリを受け取り、指定された間隔を最大距離とするように辺に頂点を追加することにより、ジオメトリを高密度化した新しいものを生成します。

構文	densify_by_distance(geometry, distance)
引数	• **geometry** - ジオメトリ（ (multi)linestrings 又は (multi)polygons を受け付けます)。 • **distance** - 出力ジオメトリの頂点間の最大間隔
例	• `geom_to_wkt(densify_by_distance(geom_from_wkt('LINESTRING(1 1, 10 1)'), 4))` → LineString (1 1, 4 1, 7 1, 10 1)

参考： **間隔による高密度化** アルゴリズム

difference

geometry1 のうち、geometry2 と交差（intersect）しない部分を表すジオメトリを返します。

構文	difference(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `geom_to_wkt( difference( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4)' ) ) )` → 'LINESTRING(4 4, 5 5)'

参考： **差分** algorithm

disjoint

ジオメトリが空間的に交差しないかどうかを判定します。ジオメトリが空間を共有していない場合は TRUE を返します。

構文	disjoint(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `disjoint( geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0 ))' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `disjoint( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'POINT(4 4)' ))` → FALSE

参考： *overlay_disjoint*

distance

2 つのジオメトリ間の最小距離（空間参照に基づく）を投影法の単位で返します。

構文	distance(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `distance( geom_from_wkt( 'POINT(4 4)' ), geom_from_wkt( 'POINT(4 8)' ) )` → 4

distance_to_vertex

インデックスで指定した点までの、ジオメトリに沿った距離を返します。

構文	distance_to_vertex(geometry, vertex)
引数	• **geometry** - ラインストリングジオメトリ • **vertex** - 頂点のインデックスで、最初の点は 0 から数える。負の値の場合には、全体の頂点数からその絶対値を引いたインデックスの頂点が選択される
例	• `distance_to_vertex(geometry:=geom_from_wkt('LineString(0 0, 10 0, 10 10)'),vertex:=1)` → 10.0

end_point

ジオメトリの最後のノードを返します。

構文	end_point(geometry)
引数	• **geometry** - ジオメトリオブジェクト
例	• `geom_to_wkt(end_point(geom_from_wkt('LINESTRING(4 0, 4 2, 0 2)')))` → 'Point (0 2)'

参考： **特定の点を抽出** アルゴリズム

exif_geotag

画像ファイルの exif ジオタグからポイントジオメトリを作成します。

構文	exif_geotag(path)
引数	• **path** - 画像ファイルのパスまたはマップレイヤの値。マップレイヤの値が指定された場合、そのレイヤーのファイルソースが使用されます。
例	• `geom_to_wkt(exif_geotag('/my/photo.jpg'))` → 'Point (2 4)'

extend

ラインストリングジオメトリの始点と終点を指定量だけ延長します。最初と最後のセグメントを、その方向を使用して延長します。距離の単位はこのジオメトリの空間参照系を使います。

構文	extend(geometry, start_distance, end_distance)
引数	• **geometry** - （マルチ）ラインストリングジオメトリ • **start_distance** - 始点の延長距離 • **end_distance** - 終点の延長距離
例	• `geom_to_wkt(extend(geom_from_wkt('LineString(0 0, 1 0, 1 1)'),1, 2))` → 'LineString (-1 0, 1 0, 1 3)' • `geom_to_wkt(extend(geom_from_wkt('MultiLineString((0 0, 1 0, 1 1), (2 2, 0 2, 0 5))'),1,2))` → 'MultiLineString ((-1 0, 1 0, 1 3),(3 2, 0 2, 0 7))'

参考： **線の延長** アルゴリズム

exterior_ring

ポリゴンジオメトリの外側のリングを表すラインストリングを返します。ジオメトリがポリゴンではない場合には、結果は NULL になります。

構文	exterior_ring(geometry)
引数	• **geometry** - ポリゴンジオメトリ
例	• `geom_to_wkt(exterior_ring(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),( 0.1 0.1, 0.1 0.2, 0.2 0.2, 0.2, 0.1, 0.1 0.1))')))` → 'LineString (-1 -1, 4 0, 4 2, 0 2, -1 -1)'

extrude

（マルチ）カーブまたは（マルチ）ラインストリングジオメトリを、x と y で指定した押し出し量だけ並行移動させてできるポリゴンを返します。

構文	extrude(geometry, x, y)
引数	• **geometry** - カーブジオメトリまたはラインストリングジオメトリ • **x** - x 方向の押し出し量の数値 • **y** - y 方向の押し出し量の数値
例	• `geom_to_wkt(extrude(geom_from_wkt('LineString(1 2, 3 2, 4 3)'), 1, 2))` → 'Polygon ((1 2, 3 2, 4 3, 5 5, 4 4, 2 4, 1 2))' • `geom_to_wkt(extrude(geom_from_wkt('MultiLineString((1 2, 3 2), (4 3, 8 3))'), 1, 2))` → 'MultiPolygon (((1 2, 3 2, 4 4, 2 4, 1 2)),((4 3, 8 3, 9 5, 5 5, 4 3)))'

flip_coordinates

X 座標と Y 座標を入れ替えたジオメトリのコピーを返します。緯度と経度を逆にしてしまったジオメトリを修復する際に便利です。

構文	flip_coordinates(geometry)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkt(flip_coordinates(make_point(1, 2)))` → 'Point (2 1)' • `geom_to_wkt(flip_coordinates(geom_from_wkt('LineString(0 2, 1 0, 1 6)')))` → 'LineString (2 0, 0 1, 6 1)'

参考： **座標の *XY* 入れ替え** アルゴリズム

force_polygon_ccw

外側のリングは反時計回り、内側のリングは時計回りになるようにジオメトリを強制的に設定します。

構文	force_polygon_ccw(geometry)
引数	• **geometry** - ジオメトリ。ポリゴンではないジオメトリはそのまま返します。
例	• `geom_to_wkt(force_polygon_ccw(geometry:=geom_from_wkt('Polygon ((-1 -1, 0 2, 4 2, 4 0, -1 -1))')))` → 'Polygon ((-1 -1, 4 0, 4 2, 0 2, -1 -1))'

参考: *force_polygon_cw*, *force_rhr*

force_polygon_cw

外側のリングは時計回り、内側のリングは反時計回りになるようにジオメトリを強制的に設定します。

構文	force_polygon_cw(geometry)
引数	• **geometry** - ジオメトリ。ポリゴンではないジオメトリはそのまま返します。
例	• `geom_to_wkt(force_polygon_cw(geometry:=geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))')))` → 'Polygon ((-1 -1, 0 2, 4 2, 4 0, -1 -1))'

参考: *force_polygon_ccw*, *force_rhr*

force_rhr

ポリゴンで囲まれる区域が境界の右側にある、「右手ルール」を尊重するようにジオメトリを強制的に設定します。特に、外側のリングは時計回りに、内側のリングは反時計回りに向きを変えます。右手ルールの定義は文脈によって不一致があるため、代わりに明白な force_polygon_cw 関数を使用することが推奨されます。

構文	force_rhr(geometry)
引数	• **geometry** - ジオメトリ。ポリゴンではないジオメトリはそのまま返します。
例	• `geom_to_wkt(force_rhr(geometry:=geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))')))` → 'Polygon ((-1 -1, 0 2, 4 2, 4 0, -1 -1))'

参考: **右手ルールを強制** アルゴリズム、 *force_polygon_ccw*, *force_polygon_cw*

geom_from_gml

ジオメトリの GML 表現文字列からジオメトリを返します。

構文	geom_from_gml(gml)
引数	• **gml** - ジオメトリの GML 表現の文字列
例	• `geom_from_gml('<gml:LineString srsName="EPSG:4326"><gml:coordinates>4, 4 5,5 6,6</gml:coordinates></gml:LineString>')` → ラインジオメトリオブジェクト

geom_from_wkb

Well-Known Binary（WKB）表現から作成されたジオメトリを返します。

構文	geom_from_wkb(binary)
引数	• **binary** - ジオメトリの Well-Known Binary（WKB）表現（バイナリの blob）
例	• `geom_from_wkb( geom_to_wkb( make_point(4,5) ) )` → ポイントジオメトリオブジェクト

geom_from_wkt

Well-Known Text（WKT）表現から作成されたジオメトリを返します。

構文	geom_from_wkt(text)
引数	• **text** - ジオメトリの Well-Known Text（WKT）表現
例	• `geom_from_wkt( 'POINT(4 5)' )` → ジオメトリオブジェクト

geom_to_wkb

ジオメトリの Well-Known Binary（WKB）表現を返します。

構文	geom_to_wkb(geometry)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkb( $geometry )` → ジオメトリオブジェクトを含むバイナリ blob

geom_to_wkt

ジオメトリの Well-known text（WKT）表現を SRID メタデータなしで返します。

構文	geom_to_wkt(geometry, [precision=8]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **precision** - 数値精度
例	• `geom_to_wkt( make_point(6, 50) )` → 'POINT(6 50)' • `geom_to_wkt(centroid(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')))` → 'POINT(0 0.66666667)' • `geom_to_wkt(centroid(geom_from_wkt('Polygon((1 1, 0 0, -1 1, 1 1))')), 2)` → 'POINT(0 0.67)'

$geometry

現在の地物のジオメトリを返します。他の関数での処理に使用することができます。**警告：この関数は非推奨です。代わりに@geometry 変数を使用することをお勧めします。**

構文	$geometry
例	• `geom_to_wkt( $geometry )` → 'POINT(6 50)'

geometry

地物のジオメトリを返します。

構文	geometry(feature)
引数	• **feature** - 地物オブジェクト
例	• `geometry( $currentfeature )` → 現在の地物のジオメトリ。 $geometry の使用が推奨される • `geom_to_wkt( geometry( get_feature_by_id( 'streets', 1 ) ) )` → "streets" レイヤ内で id が 1 の地物のジオメトリの WKT 表現。例えば 'POINT(6 50)' • `intersects( $geometry, geometry( get_feature( 'streets', 'name', 'Main St.' ) ) )` → 現在の地物が "streets" レイヤの 'Main St.' という名前の地物と空間的に交差している場合は TRUE

geometry_n

ジオメトリコレクションから特定のジオメトリを返し、入力ジオメトリがコレクションでない場合は NULL を返します。また、マルチパートジオメトリから部分を返します。

構文	geometry_n(geometry, index)
引数	• **geometry** - ジオメトリコレクション • **index** - 戻り値となるジオメトリのインデックス。コレクションの最初のジオメトリは 1
例	• `geom_to_wkt(geometry_n(geom_from_wkt('GEOMETRYCOLLECTION(POINT(0 1), POINT(0 0), POINT(1 0), POINT(1 1))'),3))` → 'Point (1 0)'

geometry_type

ジオメトリの種類を表す文字列値（Point、Line または Polygon）を返します。

構文	geometry_type(geometry)
引数	• **geometry** - ジオメトリ
例	• `geometry_type( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 'Line' • `geometry_type( geom_from_wkt( 'MULTILINESTRING((2 5, 3 6, 4 8), (1 1, 0 0))') )` → 'Line' • `geometry_type( geom_from_wkt( 'POINT(2 5)') )` → 'Point' • `geometry_type( geom_from_wkt( 'POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))') )` → 'Polygon'

hausdorff_distance

2 つのジオメトリ間のハウスドルフ距離を返します。これは基本的に、2 つのジオメトリがどれほど類似しているか、類似していないかを測る指標であり、距離が小さいほど類似したジオメトリであることを示します。

この関数はオプションとして、稠密化比率を引数で与えて実行できます。これが指定されていない場合は、標準的なハウスドルフ距離の近似値が使用されます。この近似は有用なケースの大部分では正確か、十分に近いものです。例えば、

- ほぼ互いに平行でほぼ同じ長さのラインストリング間の類似性を計算する場合。これは線形ネットワークのマッチングで使用します。
- ジオメトリの類似性をテストする場合。

この関数のデフォルトの近似値では不十分な場合、オプションで稠密化比率を引数に指定します。この引数を指定すると、離散ハウスドルフ距離を計算する前にセグメントの稠密化が実行されます。このパラメータは、各セグメントを緻密化する比率を設定します。各セグメントはこの比率で等しい長さのサブセグメントに分割されます。稠密化比率のパラメータを小さくすれば、返される距離は真のハウスドルフ距離に近づきます。

構文	hausdorff_distance(geometry1, geometry2, [densify_fraction]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ • **densify_fraction** - 稠密化比率
例	• `hausdorff_distance( geometry1:= geom_from_wkt('LINESTRING (0 0, 2 1)'),geometry2:=geom_from_wkt('LINESTRING (0 0, 2 0)'))` → 2 • `hausdorff_distance( geom_from_wkt('LINESTRING (130 0, 0 0, 0 150)'),geom_from_wkt('LINESTRING (10 10, 10 150, 130 10)'))` → 14.142135623 • `hausdorff_distance( geom_from_wkt('LINESTRING (130 0, 0 0, 0 150)'),geom_from_wkt('LINESTRING (10 10, 10 150, 130 10)'),0.5)` → 70.0

inclination

point_a に対する point_b の傾斜角を、天頂（0 度）から天底（180 度）までの角度で返します。

構文	inclination(point_a, point_b)
引数	• **point_a** - ポイントジオメトリ • **point_b** - ポイントジオメトリ
例	• `inclination( make_point( 5, 10, 0 ), make_point( 5, 10, 5 ) )` → 0.0 • `inclination( make_point( 5, 10, 0 ), make_point( 5, 10, 0 ) )` → 90.0 • `inclination( make_point( 5, 10, 0 ), make_point( 50, 100, 0 ) )` → 90.0 • `inclination( make_point( 5, 10, 0 ), make_point( 5, 10, -5 ) )` → 180.0

interior_ring_n

ポリゴンジオメトリから指定した内側のリングを返します。ジオメトリがポリゴンではない場合、NULL を返します。

構文	interior_ring_n(geometry, index)
引数	• **geometry** - ポリゴンジオメトリ • **index** - 戻り値となる内側のリングのインデックス。最初の内側リングが 1
例	• `geom_to_wkt(interior_ring_n(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),(-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1),(-1 -1, 4 0, 4 2, 0 2, -1 -1))'),1))` → 'LineString (-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1))'

intersection

2 つのジオメトリの共有部分を表すジオメトリを返します。

構文	intersection(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `geom_to_wkt( intersection( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4)' ) ) )` → 'LINESTRING(3 3, 4 4)' • `geom_to_wkt( intersection( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'MULTIPOINT(3.5 3.5, 4 5)' ) ) )` → 'POINT(3.5 3.5)'

参考： **交差** アルゴリズム

intersects

あるジオメトリが他のジオメトリと交差しているかどうかを判定します。ジオメトリが空間的に交差している（空間の任意の部分を共有している）場合は TRUE、交差していない場合は FALSE を返します。

構文	intersects(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `intersects( geom_from_wkt( 'POINT(4 4)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `intersects( geom_from_wkt( 'POINT(4 5)' ), geom_from_wkt( 'POINT(5 5)' ) )` → FALSE

参考： *overlay_intersects*

intersects_bbox

ジオメトリのバウンディングボックスが他のジオメトリのバウンディングボックスに重なるかどうかをテストします。ジオメトリが定義されたバウンディングボックスと空間的に交差している場合は TRUE、交差していない場合は FALSE を返します。

構文	intersects_bbox(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `intersects_bbox( geom_from_wkt( 'POINT(4 5)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `intersects_bbox( geom_from_wkt( 'POINT(6 5)' ), geom_from_wkt( 'POLYGON((3 3, 4 4, 5 5, 3 3))' ) )` → FALSE

is_closed

ラインストリングが閉じている（始点と終点が一致している）場合は TRUE、閉じていない場合は FALSE を返します。ジオメトリがラインストリングでない場合、結果は NULL となります。

構文	is_closed(geometry)
引数	• **geometry** - ラインストリングジオメトリ
例	• `is_closed(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)'))` → FALSE • `is_closed(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2, 0 0)'))` → TRUE

is_empty

ジオメトリが空（座標なし）の場合は TRUE、ジオメトリが空でない場合は false、ジオメトリが存在しない場合は NULL を返します。is_empty_or_null も参照してください。

構文	is_empty(geometry)
引数	• **geometry** - ジオメトリ
例	• `is_empty(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)'))` → FALSE • `is_empty(geom_from_wkt('LINESTRING EMPTY'))` → TRUE • `is_empty(geom_from_wkt('POINT(7 4)'))` → FALSE • `is_empty(geom_from_wkt('POINT EMPTY'))` → TRUE

参考: *is_empty_or_null*

is_empty_or_null

ジオメトリが NULL か空（座標値を持たない）の場合は TRUE、それ以外なら false を返します。この関数は、式 '$geometry IS NULL or is_empty($geometry)' と同様です

構文	is_empty_or_null(geometry)
引数	• **geometry** - ジオメトリ
例	• `is_empty_or_null(NULL)` → TRUE • `is_empty_or_null(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)'))` → FALSE • `is_empty_or_null(geom_from_wkt('LINESTRING EMPTY'))` → TRUE • `is_empty_or_null(geom_from_wkt('POINT(7 4)'))` → FALSE • `is_empty_or_null(geom_from_wkt('POINT EMPTY'))` → TRUE

参考: *is_empty*, *NULL*

is_multipart

マルチタイプのジオメトリならば TRUE を返します。

構文	is_multipart(geometry)
引数	• **geometry** - ジオメトリ
例	• `is_multipart(geom_from_wkt('MULTIPOINT ((0 0),(1 1),(2 2))'))` → TRUE • `is_multipart(geom_from_wkt('POINT (0 0)'))` → FALSE

is_valid

ジオメトリが有効な場合、つまり OGC の規約に従って正しく作成された 2D ジオメトリである場合には TRUE を返します。

構文	is_valid(geometry)
引数	• **geometry** - ジオメトリ
例	• `is_valid(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2, 0 0)'))` → TRUE • `is_valid(geom_from_wkt('LINESTRING(0 0)'))` → FALSE

参考: *make_valid*, **有効性チェック** algorithm

$length

ラインストリングの長さを返します。ポリゴンの境界の長さを求める場合には、代わりに $perimeter を使用してください。この関数で計算される長さは、現在のプロジェクトの楕円体の設定と距離単位の設定の両方が影響します。例えば、プロジェクトに楕円体が設定されている場合、長さは楕円体長になり、楕円体が設定されていない場合は、長さは平面的に計算されます。

構文	$length
例	• `$length` → 42.4711

length

文字列の文字数、またはラインストリングジオメトリの長さを返します。

文字列バージョン

文字列の文字数を返します。

構文	length(string)
引数	• **string** - 文字数を数える文字列
例	• `length('hello')` → 5

ジオメトリバージョン

ラインジオメトリオブジェクトの長さを返します。計算は常にこのジオメトリの空間参照系（SRS）で平面的に計算され、長さの単位は SRS の単位と一致します。これは、プロジェクトの楕円体と距離単位設定にもとづいた楕円体計算が行われる $length 関数とは異なります。

構文	length(geometry)
引数	• **geometry** - ラインジオメトリオブジェクト
例	• `length(geom_from_wkt('LINESTRING(0 0, 4 0)'))` → 4.0

参考： *straight_distance_2d*

length3D

ラインオブジェクトのジオメトリの 3 次元的な長さを返します。ジオメトリが 3 次元ラインオブジェクトでない場合は、2 次元的な長さを返します。この計算は、常にこのジオメトリの空間参照系（SRS）で平面的に計算され、長さの単位は SRS の単位と一致します。この値は、プロジェクトの楕円体と距離単位設定にもとづいた楕円体計算が行われる $length 関数とは異なります。

構文	length3D(geometry)
引数	• **geometry** - ラインジオメトリオブジェクト
例	• `length3D(geom_from_wkt('LINESTRINGZ(0 0 0, 3 0 4)'))` → 5.0

line_interpolate_angle

ラインストリングジオメトリに沿って指定された距離にある点における、ジオメトリに平行な角度を返します。角度は度単位で、北から時計回りで測ります。

構文	line_interpolate_angle(geometry, distance)
引数	• **geometry** - ラインストリングジオメトリ • **distance** - 角度を内挿したい地点の、線に沿った距離
例	• `line_interpolate_angle(geometry:=geom_from_wkt('LineString(0 0, 10 0)'),distance:=5)` → 90.0

line_interpolate_point

ラインストリングジオメトリに沿って指定された距離にある点を返します。

構文	line_interpolate_point(geometry, distance)
引数	• **geometry** - ラインストリングジオメトリ • **distance** - 位置を内挿したい地点の、線に沿った距離
例	• `geom_to_wkt(line_interpolate_point(geometry:=geom_from_wkt('LineString(0 0, 8 0)'), distance:=5))` → 'Point (5 0)' • `geom_to_wkt(line_interpolate_point(geometry:=geom_from_wkt('LineString(0 0, 1 1, 2 0)'), distance:=2.1))` → 'Point (1.48492424 0.51507576)' • `geom_to_wkt(line_interpolate_point(geometry:=geom_from_wkt('LineString(0 0, 1 0)'), distance:=2))` → NULL

参考： **線上の等間隔点** アルゴリズム

line_locate_point

指定されたポイントジオメトリに最も近い位置に対応するラインストリング上の地点の、ラインストリングに沿った距離を返します。

構文	line_locate_point(geometry, point)
引数	• **geometry** - ラインストリングジオメトリ • **point** - ラインストリング上の最も近い点を探すポイントジオメトリ
例	• `line_locate_point(geometry:=geom_from_wkt('LineString(0 0, 10 0)'),point:=geom_from_wkt('Point(5 0)'))` → 5.0

line_merge

入力されたジオメトリのうち、接続されているラインストリングをすべて単一のラインストリングにマージしたラインストリングまたはマルチラインストリングジオメトリを返します。引数にラインストリングやマルチラインストリング以外のジオメトリが渡された場合、この関数は NULL を返します。

構文	line_merge(geometry)
引数	• **geometry** - ラインストリング/マルチラインストリングジオメトリ
例	• `geom_to_wkt(line_merge(geom_from_wkt('MULTILINESTRING((0 0, 1 1), (1 1, 2 2))')))` → 'LineString(0 0,1 1,2 2)' • `geom_to_wkt(line_merge(geom_from_wkt('MULTILINESTRING((0 0, 1 1), (11 1, 21 2))')))` → 'MultiLineString((0 0, 1 1),(11 1, 21 2)'

line_substring

（ラインの始点から測った）指定された開始距離と終了距離の間にあるライン（またはカーブ）ジオメトリの部分を返します。Z 値と M 値は既存の値から線形内挿されます。

構文	line_substring(geometry, start_distance, end_distance)
引数	• **geometry** - ラインストリングまたはカーブジオメトリ • **start_distance** - 切り出しの開始距離 • **end_distance** - 切り出しの終了距離
例	• `geom_to_wkt(line_substring(geometry:=geom_from_wkt('LineString(0 0, 10 0)'),start_distance:=2,end_distance:=6))` → 'LineString (2 0,6 0)'

参考： **ラインの一部の切り出し** アルゴリズム

m

ポイントジオメトリの m 値 (measure) を返します。

構文	m(geometry)
引数	• **geometry** - ポイントジオメトリ
例	• `m( geom_from_wkt( 'POINTM(2 5 4)' ) )` → 4

m_max

ジオメトリの m 値（measure）の最大値を返します。

構文	m_max(geometry)
引数	• **geometry** - m 値を持つジオメトリ
例	• `m_max( make_point_m( 0,0,1 ) )` → 1 • `m_max(make_line( make_point_m( 0,0,1 ), make_point_m( -1,-1,2 ), make_point_m( -2,-2,0 ) ) )` → 2

m_min

ジオメトリの m 値（measure）の最小値を返します。

構文	m_min(geometry)
引数	• **geometry** - m 値を持つジオメトリ
例	• `m_min( make_point_m( 0,0,1 ) )` → 1 • `m_min(make_line( make_point_m( 0,0,1 ), make_point_m( -1,-1,2 ), make_point_m( -2,-2,0 ) ) )` → 0

main_angle

ジオメトリを完全にカバーする最小の回転長方形の長軸の角度（北を 0 度として時計回り）を返します。

構文	main_angle(geometry)
引数	• **geometry** - ジオメトリ
例	• `main_angle(geom_from_wkt('Polygon ((321577 129614, 321581 129618, 321585 129615, 321581 129610, 321577 129614))'))` → 38.66

make_circle

円形のポリゴンを作成します。

構文	make_circle(center, radius, [segments=36]) 記号 [] は、オプションの引数を意味します。
引数	• **center** - 円の中心点 • **radius** - 円の半径 • **segments** - ポリゴンのセグメント数に関するオプション引数。デフォルト値は 36
例	• `geom_to_wkt(make_circle(make_point(10,10), 5, 4))` → 'Polygon ((10 15, 15 10, 10 5, 5 10, 10 15))' • `geom_to_wkt(make_circle(make_point(10,10,5), 5, 4))` → 'PolygonZ ((10 15 5, 15 10 5, 10 5 5, 5 10 5, 10 15 5))' • `geom_to_wkt(make_circle(make_point(10,10,5,30), 5, 4))` → 'PolygonZM ((10 15 5 30, 15 10 5 30, 10 5 5 30, 5 10 5 30, 10 15 5 30))'

make_ellipse

楕円形のポリゴンを作成します。

構文	make_ellipse(center, semi_major_axis, semi_minor_axis, azimuth, [segments=36]) 記号 [] は、オプションの引数を意味します。
引数	• **center** - 楕円の中心点 • **semi_major_axis** - 楕円の半長軸 • **semi_minor_axis** - 楕円の半短軸 • **azimuth** - 楕円の向き • **segments** - ポリゴンのセグメント数に関するオプション引数。デフォルト値は 36
例	• `geom_to_wkt(make_ellipse(make_point(10,10), 5, 2, 90, 4))` → 'Polygon ((15 10, 10 8, 5 10, 10 12, 15 10))' • `geom_to_wkt(make_ellipse(make_point(10,10,5), 5, 2, 90, 4))` → 'PolygonZ ((15 10 5, 10 8 5, 5 10 5, 10 12 5, 15 10 5))' • `geom_to_wkt(make_ellipse(make_point(10,10,5,30), 5, 2, 90, 4))` → 'PolygonZM ((15 10 5 30, 10 8 5 30, 5 10 5 30, 10 12 5 30, 15 10 5 30))'

make_line

一連のポイントジオメトリからラインジオメトリを作成します。

引数バージョン

ラインの頂点は、関数の引数として個別に指定します。

構文	make_line(point1, point2, ...)
引数	• **point** - ポイントジオメトリ（またはポイントの配列）
例	• `geom_to_wkt(make_line(make_point(2,4),make_point(3,5)))` → 'LineString (2 4, 3 5)' • `geom_to_wkt(make_line(make_point(2,4),make_point(3,5), make_point(9,7)))` → 'LineString (2 4, 3 5, 9 7)'

配列バージョン

線の頂点は、ポイントの配列として指定します。

構文	make_line(array)
引数	• **array** - ポイントの配列
例	• `geom_to_wkt(make_line(array(make_point(2,4),make_point(3,5), make_point(9,7))))` → 'LineString (2 4, 3 5, 9 7)'

make_point

x と y の値（とオプションで z 値と m 値）を指定して、ポイントジオメトリを作成します。

構文	make_point(x, y, [z], [m]) 記号 [] は、オプションの引数を意味します。
引数	• **x** - ポイントの x 座標 • **y** - ポイントの y 座標 • **z** -（オプション）ポイントの z 座標 • **m** -（オプション）ポイントの m 値
例	• `geom_to_wkt(make_point(2,4))` → 'Point (2 4)' • `geom_to_wkt(make_point(2,4,6))` → 'PointZ (2 4 6)' • `geom_to_wkt(make_point(2,4,6,8))` → 'PointZM (2 4 6 8)'

make_point_m

x, y 座標および m 値からポイントジオメトリを作成します。

構文	make_point_m(x, y, m)
引数	• **x** - ポイントの x 座標 • **y** - ポイントの y 座標 • **m** - ポイントの m 値
例	• `geom_to_wkt(make_point_m(2,4,6))` → 'PointM (2 4 6)'

make_polygon

外側のリングジオメトリおよび一連の内側リングジオメトリ（オプション）から、ポリゴンジオメトリを作成します。

構文	make_polygon(outerRing, [innerRing1], [innerRing2], ...) 記号 [] は、オプションの引数を意味します。
引数	• **outerRing** - ポリゴンの外側リングとなる閉じたラインジオメトリ • **innerRing** - （オプション）内側リングとなる閉じたラインジオメトリ
例	• `geom_to_wkt(make_polygon(geom_from_wkt('LINESTRING( 0 0, 0 1, 1 1, 1 0, 0 0 )')))` → 'Polygon ((0 0, 0 1, 1 1, 1 0, 0 0))' • `geom_to_wkt(make_polygon(geom_from_wkt('LINESTRING( 0 0, 0 1, 1 1, 1 0, 0 0 )'),geom_from_wkt('LINESTRING( 0.1 0.1, 0.1 0.2, 0.2 0.2, 0.2 0.1, 0.1 0.1 )'),geom_from_wkt('LINESTRING( 0.8 0.8, 0.8 0.9, 0.9 0.9, 0.9 0.8, 0.8 0.8 )')))` → 'Polygon ((0 0, 0 1, 1 1, 1 0, 0 0),(0.1 0.1, 0.1 0.2, 0.2 0.2, 0.2 0.1, 0.1 0.1),(0.8 0.8, 0.8 0.9, 0.9 0.9, 0.9 0.8, 0.8 0.8))'

make_rectangle_3points

３点を指定して長方形を作成します。

構文	make_rectangle_3points(point1, point2, point3, [option=0]) 記号 [] は、オプションの引数を意味します。
引数	• **point1** - 第 1 点 • **point2** - 第 2 点 • **point3** - 第 3 点 • **option** - 長方形を作成するためのオプション引数。デフォルト値は 0。値は 0（距離モード）か 1（投影モード）をとる。距離モードの場合、第二距離は第 2 点と第 3 点の距離に等しくなります。投影モードの場合、第二距離は第 3 点をセグメントまたはその延長線上に垂直投影した距離に等しくなります。
例	• `geom_to_wkt(make_rectangle_3points(make_point(0, 0), make_point(0, 5), make_point(5, 5), 0))` → 'Polygon ((0 0, 0 5, 5 5, 5 0, 0 0))' • `geom_to_wkt(make_rectangle_3points(make_point(0, 0), make_point(0, 5), make_point(5, 3), 1))` → 'Polygon ((0 0, 0 5, 5 5, 5 0, 0 0))'

make_regular_polygon

正多角形を作成します。

構文	make_regular_polygon(center, radius, number_sides, [circle=0]) 記号 [] は、オプションの引数を意味します。
引数	• **center** - 正多角形の中心点 • **radius** - 2 番目の点。内接モードの場合は、最初の頂点です。外接モードの場合は、最初の辺の中間点です。 • **number_sides** - 正多角形の辺/エッジの数 • **circle** - （オプション）正多角形を作成するモードに関する引数。デフォルトは 0。値は 0 （内接モード）または 1（外接モード）をとる
例	• `geom_to_wkt(make_regular_polygon(make_point(0,0), make_point(0,5), 5))` → 'Polygon ((0 5, 4.76 1.55, 2.94 -4.05, -2.94 -4.05, -4.76 1.55, 0 5))' • `geom_to_wkt(make_regular_polygon(make_point(0,0), project(make_point(0,0), 4.0451, radians(36)), 5))` → 'Polygon ((0 5, 4.76 1.55, 2.94 -4.05, -2.94 -4.05, -4.76 1.55, 0 5))'

make_square

対角線を指定して正方形を作成します。

構文	make_square(point1, point2)
引数	• **point1** - 対角線の第 1 点 • **point2** - 対角線の第 2 点
例	• `geom_to_wkt(make_square( make_point(0,0), make_point(5,5)))` → 'Polygon ((0 0, -0 5, 5 5, 5 0, 0 0))' • `geom_to_wkt(make_square( make_point(5,0), make_point(5,5)))` → 'Polygon ((5 0, 2.5 2.5, 5 5, 7.5 2.5, 5 0))'

make_triangle

三角形のポリゴンを作成します。

構文	make_triangle(point1, point2, point3)
引数	• **point1** - 三角形の第 1 点 • **point2** - 三角形の第 2 点 • **point3** - 三角形の第 3 点
例	• `geom_to_wkt(make_triangle(make_point(0,0), make_point(5,5), make_point(0,10)))` → 'Triangle ((0 0, 5 5, 0 10, 0 0))' • `geom_to_wkt(boundary(make_triangle(make_point(0,0), make_point(5, 5), make_point(0,10))))` → 'LineString (0 0, 5 5, 0 10, 0 0)'

make_valid

有効なジオメトリ、またはジオメトリを有効にできない場合は空のジオメトリを返します。

構文	make_valid(geometry, [method=structure], [keep_collapsed=false]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **method** - 修復アルゴリズム。「structure」または「linework」のいずれかを指定します。「linework」オプションは、すべてのリングをノード化されたラインのセットに結合し、そのラインワークから有効なポリゴンを抽出します。「structure」メソッドでは、まずすべてのリングを有効なものにし、次にシェルをマージし、シェルからホールを引いて有効な結果を生成します。ホールとシェルが正しく分類されていることを前提とします。 • **keep_collapsed** - true に設定すると、低い次元に折りたたまれた要素が保持されます。例えば、リングが折りたたまれて線になったり、線が折りたたまれて点になったりするような場合です。
例	• `geom_to_wkt(make_valid(geom_from_wkt('POLYGON((3 2, 4 1, 5 8, 3 2, 4 2))')))` → 'Polygon ((3 2, 5 8, 4 1, 3 2))' • `geom_to_wkt(make_valid(geom_from_wkt('POLYGON((3 2, 4 1, 5 8, 3 2, 4 2))'), 'linework'))` → 'GeometryCollection (Polygon ((5 8, 4 1, 3 2, 5 8)),LineString (3 2, 4 2))' • `geom_to_wkt(make_valid(geom_from_wkt('POLYGON((3 2, 4 1, 5 8))'), method:='linework'))` → 'Polygon ((3 2, 4 1, 5 8, 3 2))' • `make_valid(geom_from_wkt('LINESTRING(0 0)'))` → 空のジオメトリ

参考: *is_valid*, **ジオメトリを修復** アルゴリズム

minimal_circle

ジオメトリの最小包囲円を返します。すべてのジオメトリを包み込む最小の円を表します。

構文	minimal_circle(geometry, [segments=36]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **segments** - ポリゴンのセグメント数に関するオプション引数。デフォルト値は 36
例	• `geom_to_wkt( minimal_circle( geom_from_wkt( 'LINESTRING(0 5, 0 -5, 2 1)' ), 4 ) )` → 'Polygon ((0 5, 5 -0, -0 -5, -5 0, 0 5))' • `geom_to_wkt( minimal_circle( geom_from_wkt( 'MULTIPOINT(1 2, 3 4, 3 2)' ), 4 ) )` → 'Polygon ((3 4, 3 2, 1 2, 1 4, 3 4))'

参考： **最小包含円** アルゴリズム

nodes_to_points

入力ジオメトリのすべてのノードからなるマルチポイントジオメトリを返します。

構文	nodes_to_points(geometry, [ignore_closing_nodes=false]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリオブジェクト • **ignore_closing_nodes** -（オプション）ラインやポリゴンのリングを閉じる重複ノードを戻り値に含めないかどうかを指定します。デフォルトは false です。アウトプットのコレクションに重複ノードを含めないようにするには、true を指定します。
例	• `geom_to_wkt(nodes_to_points(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)')))` → 'MultiPoint ((0 0),(1 1),(2 2))' • `geom_to_wkt(nodes_to_points(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1))'),true))` → 'MultiPoint ((-1 -1),(4 0),(4 2),(0 2))'

参考： **頂点を抽出** アルゴリズム

num_geometries

ジオメトリコレクションに含まれるジオメトリの数、またはマルチパーツジオメトリに含まれるパーツの数を返します。入力ジオメトリがコレクションでない場合、この関数は NULL を返します。

構文	num_geometries(geometry)
引数	• **geometry** - ジオメトリコレクションまたはマルチパートジオメトリ
例	• `num_geometries(geom_from_wkt('GEOMETRYCOLLECTION(POINT(0 1), POINT(0 0), POINT(1 0), POINT(1 1))'))` → 4 • `num_geometries(geom_from_wkt('MULTIPOINT((0 1), (0 0), (1 0))'))` → 3

num_interior_rings

ポリゴンまたはジオメトリコレクション中の内側のリングの数を返します。ポリゴンでもジオメトリコレクションでもない場合、NULL を返します。

構文	num_interior_rings(geometry)
引数	• **geometry** - 入力ジオメトリ
例	• `num_interior_rings(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1),(-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1))'))` → 1

num_points

ジオメトリの頂点数を返します。

構文	num_points(geometry)
引数	• **geometry** - ジオメトリ
例	• `num_points($geometry)` → 現在の地物のジオメトリの頂点数

num_rings

ポリゴンまたはジオメトリコレクション中のリング（外側のリングを含む）の数を返します。ポリゴンでもジオメトリコレクションでもない場合、NULL を返します。

構文	num_rings(geometry)
引数	• **geometry** - 入力ジオメトリ
例	• `num_rings(geom_from_wkt('POLYGON((-1 -1, 4 0, 4 2, 0 2, -1 -1), (-0.1 -0.1, 0.4 0, 0.4 0.2, 0 0.2, -0.1 -0.1))'))` → 2

offset_curve

ラインストリングジオメトリを側面方向にオフセットさせたジオメトリを返します。距離はこのジオメトリの空間参照系の単位です。

構文	offset_curve(geometry, distance, [segments=8], [join=1], [miter_limit=2.0]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - （マルチ）ラインストリングジオメトリ • **distance** - オフセット距離。正の値はラインの左側にバッファされ、負の値は右にバッファされます • **segments** - 継ぎ目スタイルに round を使用する場合に、円の四分の一を表現するのに使用するセグメントの数。数値が大きいほど、より多数のノードからなる、滑らかなラインが生成されます。 • **join** - 角の継ぎ目スタイル。1 は round、2 は miter 、3 は bevel • **miter_limit** - 非常に鋭い角に使用される miter 比の制限（miter 継ぎ目の場合のみ）
例	• `offset_curve($geometry, 10.5)` → 左に 10.5 単位オフセットしたライン • `offset_curve($geometry, -10.5)` → 右に 10.5 単位オフセットしたライン • `offset_curve($geometry, 10.5, segments:=16, join:=1)` → 左に 10.5 単位オフセットしたラインで、より多数のセグメントを使用した滑らかなカーブ • `offset_curve($geometry, 10.5, join:=3)` → 左に 10.5 単位オフセットしたラインで、bebel 継ぎ目を使用

参考： **オフセット線** アルゴリズム

order_parts

与えられた基準によってマルチジオメトリのパートを並べ替えます。

構文	order_parts(geometry, orderby, [ascending=true]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - マルチタイプのジオメトリ • **orderby** - 順序の基準を定義する式 • **ascending** - ブール値。True ならば昇順で、False ならば降順で並べる
例	• `geom_to_wkt(order_parts(geom_from_wkt('MultiPolygon (((1 1, 5 1, 5 5, 1 5, 1 1)),((1 1, 9 1, 9 9, 1 9, 1 1)))'), 'area($geometry)', False))` → 'MultiPolygon (((1 1, 9 1, 9 9, 1 9, 1 1)),((1 1, 5 1, 5 5, 1 5, 1 1)))' • `geom_to_wkt(order_parts(geom_from_wkt('LineString(1 2, 3 2, 4 3)'), '1', True))` → 'LineString(1 2, 3 2, 4 3)'

oriented_bbox

入力ジオメトリに対して最小となる回転バウンディングボックスのジオメトリを返します。

構文	oriented_bbox(geometry)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkt( oriented_bbox( geom_from_wkt( 'MULTIPOINT(1 2, 3 4, 3 2)' ) ) )` → 'Polygon ((3 2, 3 4, 1 4, 1 2, 3 2))'

参考： **最小の回転長方形** アルゴリズム

overlaps

あるジオメトリがもう一つのジオメトリにオーバーラップするかどうかをテストします。ジオメトリが空間を共有していて、同じ次元であり、かつ、互いが完全に含まれない場合に TRUE を返します。

構文	overlaps(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `overlaps( geom_from_wkt( 'LINESTRING(3 5, 4 4, 5 5, 5 3)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `overlaps( geom_from_wkt( 'LINESTRING(0 0, 1 1)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → FALSE

overlay_contains

現在の地物が対象レイヤの少なくとも 1 つの地物を空間的に含んでいるかどうか、または、現在の地物に含まれる対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Contains" 述語については、PostGIS の ST_Contains 関数の説明を参照してください。

構文	overlay_contains(layer, [expression], [filter], [limit], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** -（オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** -（オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** -（オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_contains('regions')` → 現在の地物が regions の地物を空間的に含む場合に TRUE • `overlay_contains('regions', filter:= population > 10000)` → 現在の地物が population が 10000 より大きい regions の地物を空間的に含む場合に TRUE • `overlay_contains('regions', name)` → 現在の地物に含まれる regions の地物の名前の配列 • `array_to_string(overlay_contains('regions', name))` → 現在の地物に含まれる regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_contains(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物に含まれる、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_contains(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物に含まれる regions の地物のジオメトリ（WKT 形式、上限 2 個）の配列

参考： *contains* 、 **配列関数** 、 **場所による選択** アルゴリズム

overlay_crosses

現在の地物が対象レイヤの少なくとも 1 つの地物と空間的にクロスしているかどうか、または、現在の地物とクロスする対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Crosses" 述語については、PostGIS の ST_Crosses 関数の説明を参照してください。

構文	overlay_crosses(layer, [expression], [filter], [limit], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** -（オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** -（オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** -（オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_crosses('regions')` → 現在の地物が region と空間的にクロスする場合に TRUE • `overlay_crosses('regions', filter:= population > 10000)` → 現在の地物が population が 10000 を超える region と空間的にクロスする場合に TRUE • `overlay_crosses('regions', name)` → 現在の地物とクロスする regions の地物の名前の配列 • `array_to_string(overlay_crosses('regions', name))` → 現在の地物とクロスする regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_crosses(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物とクロスする、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_crosses(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物とクロスする regions の地物のジオメトリ（WKT 形式、上限 2 個）の配列

参考： *crosses* 、 **配列関数** 、 **場所による選択** アルゴリズム

overlay_disjoint

現在の地物が対象レイヤのすべての地物と空間的に非接続であるかどうか、または、現在の地物と非接続な対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Disjoint" 述語については、PostGIS の ST_Disjoint 関数の説明を参照してください。

構文	overlay_disjoint(layer, [expression], [filter], [limit], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** - （オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** - （オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** - （オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_disjoint('regions')` → 現在の地物が regions のすべての地物と空間的に非接続である場合に TRUE • `overlay_disjoint('regions', filter:= population > 10000)` → 現在の地物が population が 10000 より大きい regions の地物すべてと空間的に非接続である場合に TRUE • `overlay_disjoint('regions', name)` → 現在の地物と空間的に非接続な regions の地物の名前の配列 • `array_to_string(overlay_disjoint('regions', name))` → 現在の地物と空間的に非接続な regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_disjoint(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物と空間的に非接続な、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_disjoint(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物と空間的に非接続な regions の地物のジオメトリ（WKT 形式、上限 2 個）の配列

参考： *disjoint* 、 **配列関数** 、 **場所による選択** アルゴリズム

overlay_equals

現在の地物が対象レイヤの少なくとも 1 つの地物と空間的に同じかどうか、または、現在の地物と空間的に同じ対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Equals" 述語については、PostGIS の ST_Equals 関数の説明を参照してください。

構文	overlay_equals(layer, [expression], [filter], [limit], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** -（オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** -（オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** -（オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_equals('regions')` → 現在の地物が regions と空間的に等しい場合に TRUE • `overlay_equals('regions', filter:= population > 10000)` → 現在の地物が population が 10000 より大きい regions の地物と空間的に同じ場合に TRUE • `overlay_equals('regions', name)` → 現在の地物と空間的に同じ regions の地物の名前の配列 • `array_to_string(overlay_equals('regions', name))` → 現在の地物と空間的に同じ regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_equals(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物と空間的に同じ、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_equals(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物と空間的に同じ regions の地物のジオメトリ（WKT 形式、上限 2 個）の配列

参考： **配列関数** 、 **場所による選択** アルゴリズム

overlay_intersects

現在の地物が対象レイヤの少なくとも 1 つの地物と空間的に交差しているかどうか、または、現在の地物と交差する対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Intersects" 述語については、PostGIS の ST_Intersects 関数の説明を参照してください。

構文	overlay_intersects(layer, [expression], [filter], [limit], [cache=false], [min_overlap], [min_inscribed_circle_radius], [return_details], [sort_by_intersection_size]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** -（オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** -（オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** -（オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です） • **min_overlap** -（オプション）除外フィルタを定義します: – ポリゴンの場合は、現在の地物の平方単位による交差の最小面積。交差が複数のポリゴンで構成されている場合、そのうち少なくとも 1 つのポリゴンの面積がこの値以上であれば、交差が返されます。 – ラインの場合、現在の地物の単位による最小の長さ。交差が複数のラインになる場合、そのうち少なくとも 1 つのラインの長さがこの値以上であれば、交差が返されます。 • **min_inscribed_circle_radius** - オプションの除外フィルター（ポリゴンのみ）を定義します：現在の地物の単位による交差の最大内接円の最小半径。交差が複数のポリゴンからなる場合、ポリゴンの少なくとも 1 つが最大内接円の半径がこの値以上であれば、交差が返されます。 この関数の基礎となる GEOS の述語については、PostGIS ST_MaximumInscribedCircle 関数の説明を参照してください。 この引数は GEOS >= 3.9 を必要とします。 • **return_details** - true に設定すると、地物'id'、式の'result'、'overlap' の値を含むマップのリストを返します（キー名は引用符で囲む）。また、対象レイヤーがポリゴンの場合、最大内接円の'radius' も返されます。expression パラメータと併用した場合のみ有効 • **sort_by_intersection_size** - 式と一緒に使用した場合のみ有効です。この値を 'des' に設定すると、オーバーラップ値で降順に、 'asc' に設定すると昇順に結果を返します。
例	• `overlay_intersects('regions')` → 現在の地物が regions の地物と空間的に交差する場合に TRUE • `overlay_intersects('regions', filter:= population > 10000)` → 現在の地物が population が 10000 より大きい regions の地物と空間的に交差する場合に TRUE • `overlay_intersects('regions', name)` → 現在の地物と交差する regions の地物の名前の配列 • `array_to_string(overlay_intersects('regions', name))` → 現在の地物と交差する regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_intersects(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物と交差する、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_intersects(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物と交差する regions

参考： *intersects* 、 **配列関数** 、 **場所による選択** アルゴリズム

overlay_nearest

現在の地物が対象レイヤの地物と指定された距離以内にあるかどうか、または、現在の地物から指定された距離以内にある対象レイヤの地物に関する QGIS 式の結果の配列を返します。

注意: この関数は、巨大なレイヤに対しては遅く、大量のメモリを消費することがあります。

構文	overlay_nearest(layer, [expression], [filter], [limit=1], [max_distance], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - 対象レイヤ • **expression** - （オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** - （オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、 対象レイヤの全ての地物をチェックします。 • **limit** - （オプション）マッチする地物の上限数。設定されていない場合、最近傍の地物のみを返します。-1 を設定した場合、マッチする全ての地物を返します。 • **max_distance** - （オプション）マッチする地物の検索を制限するための距離。設定されていない場合、対象レイヤの全ての地物を検索に使用します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_nearest('airports')` → "airports" レイヤに少なくとも 1 つ地物がある場合に TRUE • `overlay_nearest('airports', max_distance:= 5000)` → 現在の地物から距離 5000 地図単位以内に airports の地物がある場合に TRUE • `overlay_nearest('airports', name)` → 現在の地物から最も近い airports の地物の名前の配列 • `array_to_string(overlay_nearest('airports', name))` → 現在の地物から最も近い airports の地物の名前の文字列 • `overlay_nearest(layer:='airports', expression:= name, max_distance:= 5000)` → 現在の地物から距離 5000 地図単位以内にある airports の地物のうち、最も近いものの名前の配列 • `overlay_nearest(layer:='airports', expression:="name", filter:= "Use"='Civilian', limit:=3)` → 現在の地物から最も近い最大 3 つの民間空港の名前を距離でソートした配列 • `overlay_nearest(layer:='airports', expression:="name", limit:= -1, max_distance:= 5000)` → 現在の地物から距離 5000 地図単位以内にあるすべての airports の地物の名前を距離順でソートした配列

参考： **配列関数** 、 **属性の最近傍結合** アルゴリズム

overlay_touches

現在の地物が対象レイヤの少なくとも 1 つの地物と空間的に接触しているかどうか、または、現在の地物と接触する対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Touches" 述語については、PostGIS の ST_Touches 関数の説明を参照してください。

構文	overlay_touches(layer, [expression], [filter], [limit], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** - （オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** - （オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** - （オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_touches('regions')` → 現在の地物が regions の地物と空間的に接触する場合に TRUE • `overlay_touches('regions', filter:= population > 10000)` → 現在の地物が population が 10000 より大きい regions の地物と空間的に接触する場合に TRUE • `overlay_touches('regions', name)` → 現在の地物と接触する regions の地物の名前の配列 • `array_to_string(overlay_touches('regions', name))` → 現在の地物と接触する regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_touches(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物と接触する、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_touches(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物と接触する regions の地物のジオメトリ（WKT 形式、上限 2 個）の配列

参考： *touches* 、 **配列関数** 、 **場所による選択** アルゴリズム

overlay_within

現在の地物が対象レイヤの少なくとも 1 つの地物に空間的に含まれているかどうか、または、現在の地物を含んでいる対象レイヤの地物に関する QGIS 式の結果の配列を返します。

この関数の基礎となる GEOS の "Within" 述語については、PostGIS の ST_Within 関数の説明を参照してください。

構文	overlay_within(layer, [expression], [filter], [limit], [cache=false]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - オーバーレイがチェックされるレイヤ • **expression** - （オプション）対象レイヤの地物を評価する式。設定されていない場合には、この関数は一つ以上のマッチがあるかどうかを示すブール値を返します。 • **filter** - （オプション）チェック対象地物をフィルタリングするための式。設定されていない場合、全ての地物をチェックします。 • **limit** - （オプション）マッチする地物の上限数。設定されていない場合、マッチする全ての地物を返します。 • **cache** - この引数を true に設定すると、ローカルな空間インデックスを作成します（特に遅いデータプロバイダで作業していない限りは、ほとんどの場合でこれは不要です）
例	• `overlay_within('regions')` → 現在の地物が regions の地物によって空間的に含まれる場合に TRUE • `overlay_within('regions', filter:= population > 10000)` → 現在の地物が population が 10000 より大きい regions の地物によって空間的に含まれる場合に TRUE • `overlay_within('regions', name)` → 現在の地物を含む regions の地物の名前の配列 • `array_to_string(overlay_within('regions', name))` → 現在の地物を含む regions の地物の名前をコンマ区切りで並べた文字列 • `array_sort(overlay_within(layer:='regions', expression:="name", filter:= population > 10000))` → 現在の地物を含む、population が 10000 よりも大きな値を持つ regions の地物の名前をソートした配列 • `overlay_within(layer:='regions', expression:= geom_to_wkt($geometry), limit:=2)` → 現在の地物を含む regions の地物のジオメトリ（WKT 形式、上限 2 個）の配列

参考： *within* 、 **配列関数** 、 **場所による選択** アルゴリズム

$perimeter

現在の地物の周長を返します。この関数で計算される周長は、現在のプロジェクトの楕円体の設定と距離単位の設定の両方が反映されます。例えば、プロジェクトに回転楕円体が設定されている場合は、計算された周長は楕円体長となり、楕円が設定されていない場合には、計算された周長は平面上の長さになります。

構文	$perimeter
例	• `$perimeter` → 42

perimeter

ポリゴンオブジェクトのジオメトリの周長を返します。周長はこのジオメトリの空間参照系（SRS）で常に平面的に計算され、周長の単位は空間参照系の単位と一致します。この周長は、プロジェクトの楕円体と距離単位設定にもとづいた楕円体計算が行われる $perimeter 関数による面積とは異なります。

構文	perimeter(geometry)
引数	• **geometry** - ポリゴンジオメトリオブジェクト
例	• `perimeter(geom_from_wkt('POLYGON((0 0, 4 0, 4 2, 0 2, 0 0))'))` → 12.0

point_n

ジオメトリの特定のノードを返します。

構文	point_n(geometry, index)
引数	• **geometry** - ジオメトリオブジェクト • **index** - ノードのインデックス。1 は最初のノードです。値が負の場合には、全体の頂点数からその絶対値を引いたインデックスの頂点が選択される
例	• `geom_to_wkt(point_n(geom_from_wkt('POLYGON((0 0, 4 0, 4 2, 0 2, 0 0))'),2))` → 'Point (4 0)'

参考： **特定の点を抽出** アルゴリズム

point_on_surface

ジオメトリの表面上に存在することが保証される点を返します。

構文	point_on_surface(geometry)
引数	• **geometry** - ジオメトリ
例	• `point_on_surface($geometry)` → ポイントジオメトリ

参考： **内部保証点（*point on surface*）** アルゴリズム

pole_of_inaccessibility

ポリゴンの境界から最も離れた内部点である、ポリゴンの近似的な到達不能極を計算します。この関数は、指定された許容範囲内で真の到達不能極をみつけることが保証されている反復的手法である 'polylabel' アルゴリズム（Vladimir Agafonkin, 2016）を使用します。許容範囲を厳しくすると、より多くの反復計算が必要となり、計算時間がかかります。

構文	pole_of_inaccessibility(geometry, tolerance)
引数	• **geometry** - ジオメトリ • **tolerance** - 真の極位置と関数戻り値のポイントの間に許容する最大距離
例	• `geom_to_wkt(pole_of_inaccessibility( geom_from_wkt('POLYGON((0 1, 0 9, 3 10, 3 3, 10 3, 10 1, 0 1))'), 0.1))` → 'Point(1.546875 2.546875)'

参考： **到達不能極（*PIA*）** アルゴリズム

project

距離と、方向（方位角）および仰角をラジアン単位で指定して、始点から投影された点を返します。

構文	project(point, distance, azimuth, [elevation]) 記号 [] は、オプションの引数を意味します。
引数	• **point** - 起点 • **distance** - 投影距離 • **azimuth** - 北を 0 として時計回りで測ったラジアン単位の方位角 • **elevation** - ラジアン単位の傾斜角
例	• `geom_to_wkt(project(make_point(1, 2), 3, radians(270)))` → 'Point(-2, 2)'

参考： **投影点（デカルト座標）** アルゴリズム

relate

2 つのジオメトリ間の関係の Dimensional Extended 9 Intersection Model（DE-9IM）表現を返します。

リレーションバージョン

2 つのジオメトリ間の関係の Dimensional Extended 9 Intersection Model（DE-9IM）表現を返します。

構文	relate(geometry, geometry)
引数	• **geometry** - ジオメトリ • **geometry** - ジオメトリ
例	• `relate( geom_from_wkt( 'LINESTRING(40 40,120 120)' ), geom_from_wkt( 'LINESTRING(40 40,60 120)' ) )` → 'FF1F00102'

パターンマッチバージョン

2 つのジオメトリ間の DE-9IM 関係が、指定パターンに一致するかどうかをテストします。

構文	relate(geometry, geometry, pattern)
引数	• **geometry** - ジオメトリ • **geometry** - ジオメトリ • **pattern** - マッチさせる DE-9IM パターン
例	• `relate( geom_from_wkt( 'LINESTRING(40 40,120 120)' ), geom_from_wkt( 'LINESTRING(40 40,60 120)' ), '**1F001**' )` → TRUE

reverse

頂点の順序を逆にすることによってラインストリングの方向を反転させます。

構文	reverse(geometry)
引数	• **geometry** - ジオメトリ
例	• `geom_to_wkt(reverse(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)')))` → 'LINESTRING(2 2, 1 1, 0 0)'

参考： **線の向きの反転** アルゴリズム

rotate

回転したジオメトリを返します。計算はジオメトリの空間参照系で行われます。

構文	rotate(geometry, rotation, [center=NULL], [per_part=false]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **rotation** - 時計回りの度単位の回転量 • **center** - 回転の中心。指定しない場合はバウンディングボックスの中心を使用します • **per_part** - パーツごとに回転を適用します。true を指定すると、入力ジオメトリがマルチパートで、明示的な回転中心点が指定されていない場合、各パーツのバウンディングボックスの中心を中心に回転が適用されます。
例	• `rotate($geometry, 45, make_point(4, 5))` → 点（4, 5）を中心に時計回りに 45 度回転させたジオメトリ • `rotate($geometry, 45)` → バウンディングボックスの中央を中心に時計回りに 45 度回転させたジオメトリ

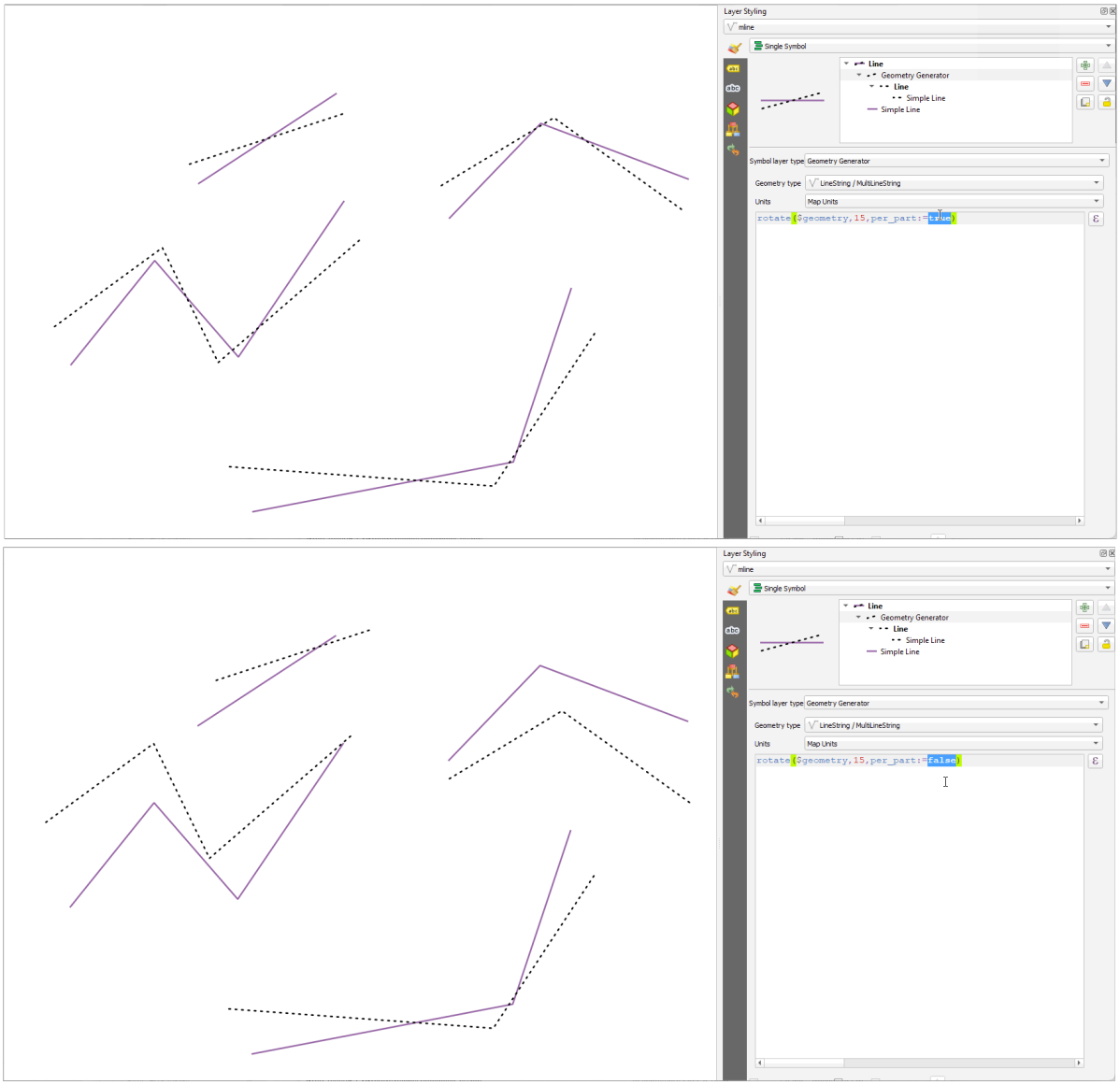


図 13.7: 地物の回転

roundness

ポリゴン形状が円にどれだけ近いかを計算します。この関数は、ポリゴン形状が完全な円である場合に TRUE を、完全に平坦である場合に 0 を返します。

構文	roundness(geometry)
引数	• **geometry** - ポリゴン
例	• `round(roundness(geom_from_wkt('POLYGON(( 0 0, 0 1, 1 1, 1 0, 0 0))')), 3)` → 0.785 • `round(roundness(geom_from_wkt('POLYGON(( 0 0, 0 0.1, 1 0.1, 1 0, 0 0))')), 3)` → 0.260

Further reading: **丸み係数** アルゴリズム

scale

拡大縮小したジオメトリを返します。計算はジオメトリの空間参照系で行われます。

構文	scale(geometry, x_scale, y_scale, [center]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **scale_x** - x 軸方向スケーリング係数 • **scale_y** - y 軸方向スケーリング係数 • **center** - 拡大縮小の中心。指定しない場合はジオメトリのバウンディングボックスの中心を使用します
例	• `scale($geometry, 2, 0.5, make_point(4, 5))` → (4, 5) の点を中心に、水平方向に 2 倍、垂直方向に半分にスケールされたジオメトリ • `scale($geometry, 2, 0.5)` → そのバウンディングボックスの中央を中心にジオメトリを水平方向に 2 倍、垂直方向に半分にする

segments_to_lines

入力ジオメトリのすべてのセグメントからなるマルチラインジオメトリを返します。

構文	segments_to_lines(geometry)
引数	• **geometry** - ジオメトリオブジェクト
例	• `geom_to_wkt(segments_to_lines(geom_from_wkt('LINESTRING(0 0, 1 1, 2 2)')))` → 'MultiLineString ((0 0, 1 1),(1 1, 2 2))'

参考： **線をセグメントに分解** アルゴリズム

shared_paths

2 つの入力ジオメトリが共有するパスを含むコレクションを返します。同じ方向に進むものはコレクションの最初の要素に、反対方向に進むものは 2 番目の要素にあります。パスそのものは、最初のジオメトリの方向で与えられます。

構文	shared_paths(geometry1, geometry2)
引数	• **geometry1** - ラインストリング/マルチラインストリングジオメトリ • **geometry2** - ラインストリング/マルチラインストリングジオメトリ
例	• `geom_to_wkt(shared_paths(geom_from_wkt('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),(51 150,101 150,76 175,51 150))'), geom_from_wkt('LINESTRING(151 100,126 156.25,126 125,90 161, 76 175)'))` → 'GeometryCollection (MultiLineString ((126 156.25, 126 125),(101 150, 90 161),(90 161, 76 175)),MultiLineString EMPTY)' • `geom_to_wkt(shared_paths(geom_from_wkt('LINESTRING(76 175,90 161, 126 125,126 156.25,151 100)'),geom_from_wkt('MULTILINESTRING((26 125,26 200,126 200,126 125,26 125),(51 150,101 150,76 175,51 150))'))` → 'GeometryCollection (MultiLineString EMPTY,MultiLineString ((76 175, 90 161),(90 161, 101 150),(126 125, 126 156.25)))'

shortest_line

geometry1 と geometry2 を結合する最短のラインを返します。向きは、geometry1 を始点とし、geometry2 を終点とします。

構文	shortest_line(geometry1, geometry2)
引数	• **geometry1** - 始点側のジオメトリ • **geometry2** - 終点側のジオメトリ
例	• `geom_to_wkt(shortest_line(geom_from_wkt('LINESTRING (20 80, 98 190, 110 180, 50 75 )'),geom_from_wkt('POINT(100 100)')))` → 'LineString(73.0769 115.384, 100 100)'

simplify

距離の閾値を使用してノードを削除し、ジオメトリを単純化します（Douglas Peucker アルゴリズム）。このアルゴリズムは、ジオメトリの大きな変動は維持し、ほぼ直線のセグメント上にある頂点を減らします。

構文	simplify(geometry, tolerance)
引数	• **geometry** - ジオメトリ • **tolerance** - 除去される点を決定するための、直線セグメントからの最大偏差
例	• `geom_to_wkt(simplify(geometry:=geom_from_wkt('LineString(0 0, 5 0.1, 10 0)'),tolerance:=5))` → 'LineString(0 0, 10 0)'

参考： **ジオメトリの簡素化** アルゴリズム

simplify_vw

面積の閾値を使ってノードを削除し、ジオメトリを単純化します（Visvalingam-Whyatt アルゴリズム）。このアルゴリズムは、細長いスパイクやほぼ直線上にあるセグメントなど、非常に小さな面積を生み出す頂点を削除します。

構文	simplify_vw(geometry, tolerance)
引数	• **geometry** - ジオメトリ • **tolerance** -除去されるノードを決定するための、ノードによって形成される面積の最小値
例	• `geom_to_wkt(simplify_vw(geometry:=geom_from_wkt('LineString(0 0, 5 0, 5.01 10, 5.02 0, 10 0)'),tolerance:=5))` → 'LineString(0 0, 10 0)'

参考： **ジオメトリの簡素化** アルゴリズム

single_sided_buffer

ラインストリングの片側だけにバッファを作ったジオメトリを返します。距離単位はこのジオメトリの空間参照系の単位です。

構文	single_sided_buffer(geometry, distance, [segments=8], [join=1], [miter_limit=2.0]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - （マルチ）ラインストリングジオメトリ • **distance** - バッファ距離。正の値はラインの左側に、負の値はラインの右側にバッファを作成します • **segments** - 継ぎ目スタイルに round を使用する場合に、円の四分の一を表現するのに使用するセグメントの数。数値が大きいほど、より多数のノードからなる、滑らかなバッファが生成されます。 • **join** - 角の継ぎ目スタイル。1 は round、2 は miter 、3 は bevel • **miter_limit** - 非常に鋭い角に使用される miter 比の制限（miter 継ぎ目の場合のみ）
例	• `single_sided_buffer($geometry, 10.5)` → ラインから左に 10.5 単位のバッファ • `single_sided_buffer($geometry, -10.5)` → ラインから右に 10.5 単位のバッファ • `single_sided_buffer($geometry, 10.5, segments:=16, join:=1)` → ラインから左に 10.5 単位のバッファで、より多数のセグメントを使用した滑らかなバッファ • `single_sided_buffer($geometry, 10.5, join:=3)` → ラインから左に 10.5 単位のバッファで、bebel 継ぎ目を使用

参考： **片側バッファ** アルゴリズム

sinuosity

曲線の湾曲率、すなわち曲線の両端の（平面的な）直線距離に対する曲線長さの比率を返します。

構文	sinuosity(geometry)
引数	• **geometry** - 入力の曲線（円形ストリング、ラインストリング）
例	• `round(sinuosity(geom_from_wkt('LINESTRING(2 0, 2 2, 3 2, 3 3)')), 3)` → 1.265 • `sinuosity(geom_from_wkt('LINESTRING( 3 1, 5 1)'))` → 1.0

smooth

ノードを追加して角を丸め、ジオメトリを滑らかにしたものを返します。入力ジオメトリに Z 値または M 値が含まれている場合には、これらの値も同様に平滑化され、出力ジオメトリは入力ジオメトリと同じ次元を保ちます。

構文	smooth(geometry, [iterations=1], [offset=0.25], [min_length=-1], [max_angle=180]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **iterations** - 適用するスムージングの反復回数。数値が大きいほど、より滑らかになりますが、ジオメトリは複雑になります。 • **offset** - 0 から 0.5 までの値で、滑らかにしたジオメトリが元のジオメトリにどれくらい従っているかを制御します。値を小さくすると元ジオメトリによりきつく追従したスムージングとなり、値を大きくすると追従の緩いスムージングとなります。 • **min_length** - スムージングを適用するセグメントの長さの下限。このパラメータは、ジオメトリの短いセグメントに余分なノードを追加することを避けるために使用されます。 • **max_angle** - 平滑化を適用するノードの最大角度（0〜180）。最大角度を小さくすると、鋭角を維持することができます。例えば、80 度とすると、ジオメトリの直角部分は維持されます。
例	• `geom_to_wkt(smooth(geometry:=geom_from_wkt('LineString(0 0, 5 0, 5 5)'),iterations:=1,offset:=0.2,min_length:=-1,max_angle:=180))` → 'LineString (0 0, 4 0, 5 1, 5 5)'

参考： **スムージング** アルゴリズム

square_wave

ジオメトリの境界に沿って方形波/矩形波を構築します。

構文	square_wave(geometry, wavelength, amplitude, [strict=False]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **wavelength** - 方形波の波長 • **amplitude** - 矩形波の振幅 • **strict** - デフォルトでは、波長の引数は「最大波長」として扱われ、実際の波長はジオメトリの境界に沿って正確な数の矩形波が作成されるように動的に調整されます。strict 引数を true に設定すると、波長が正確に使用され、最終的な波形に不完全なパターンが使用される可能性があります。
例	• `square_wave(geom_from_wkt('LineString(0 0, 10 0)'), 3, 1)` → ラインストリングに沿った波長 3、振幅 1 の方形波

square_wave_randomized

ジオメトリの境界に沿ってランダムな方形波／矩形波を構築します。

構文	square_wave_randomized(geometry, min_wavelength, max_wavelength, min_amplitude, max_amplitude, [seed=0]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **min_wavelength** - 最小の波長 • **max_wavelength** - 最大の波長 • **min_amplitude** - 最小の振幅 • **max_amplitude** - 最大の振幅 • **seed**は、波を生成するための乱数の根を指定します。seed を 0 にすると、完全にランダムな波のセットが生成されます。
例	• `square_wave_randomized(geom_from_wkt('LineString(0 0 0, 10 0)'), 2, 3, 0.1, 0.2)` → ラインストリングに沿った、波長が 2〜3、振幅が 0.1〜0.2 のランダムなサイズの方形波

start_point

ジオメトリの最初のノードを返します。

構文	start_point(geometry)
引数	• **geometry** - ジオメトリオブジェクト
例	• `geom_to_wkt(start_point(geom_from_wkt('LINESTRING(4 0, 4 2, 0 2)')))` → 'Point (4 0)'

参考： **特定の点を抽出** アルゴリズム

straight_distance_2d

ジオメトリの始点と終点の間の直線距離（ユークリッド距離）を返します。ジオメトリは曲線（円形ストリングまたはラインストリング）でなければなりません。

構文	straight_distance_2d(geometry)
引数	• **geometry** - ジオメトリ
例	• `straight_distance_2d(geom_from_wkt('LINESTRING(1 0, 1 1)'))` → 1 • `round(straight_distance_2d(geom_from_wkt('LINESTRING(1 4, 3 5, 5 0)')), 3)` → 5.657

参考： *length*

sym_difference

2 つのジオメトリの、交差しない部分を表すジオメトリを返します。

構文	sym_difference(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `geom_to_wkt( sym_difference( geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ), geom_from_wkt( 'LINESTRING(3 3, 8 8)' ) ) )` → 'LINESTRING(5 5, 8 8)'

参考： **対称差** アルゴリズム

tapered_buffer

ラインジオメトリに沿って、バッファの直径が均等に変化していくバッファを作成します。

構文	tapered_buffer(geometry, start_width, end_width, [segments=8]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - 入力ジオメトリ。（マルチ）ラインジオメトリである必要があります。 • **start_width** - 始点のバッファ幅 • **end_width** - 終点のバッファ幅 • **segments** - バッファの四分円を近似するセグメント数
例	• `tapered_buffer(geometry:=geom_from_wkt('LINESTRING(1 2, 4 2)'), start_width:=1,end_width:=2,segments:=8)` → ラインストリングジオメトリに沿って、直径 1 で始まり、直径 2 で終わるテーパードバッファ

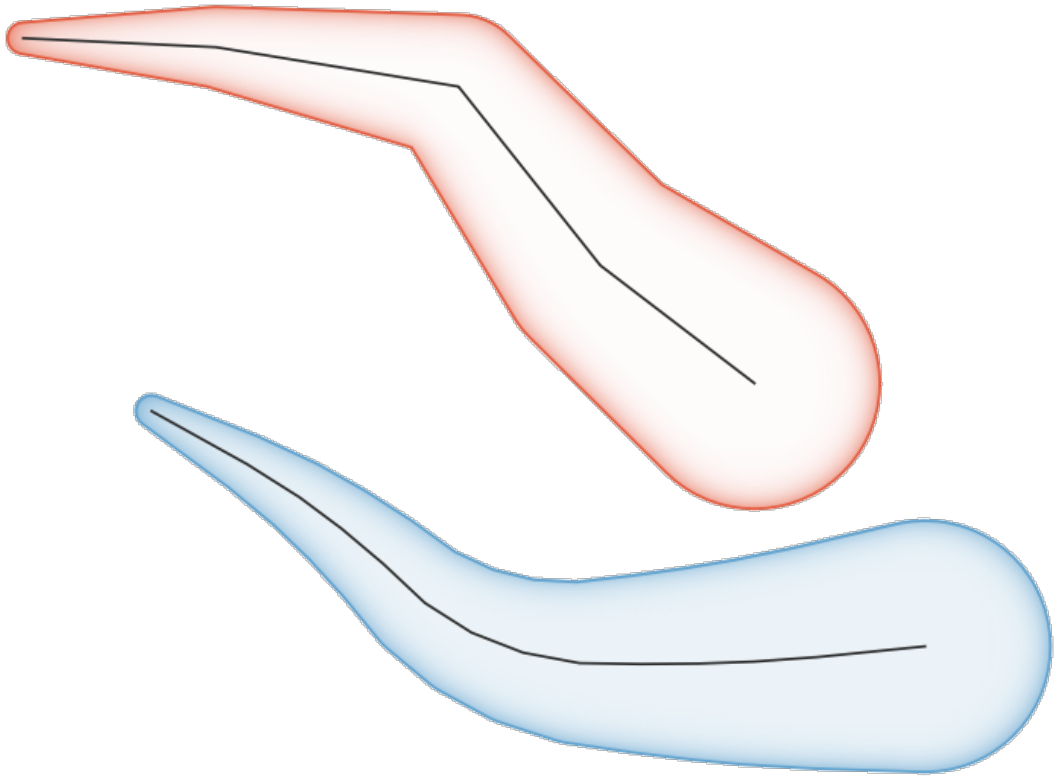


図 13.8: ライン地物に対するテーパードバッファ

参考： **テイパードバッファ** アルゴリズム

touches

あるジオメトリがもう一つのジオメトリに接するかどうかをテストします。ジオメトリが共通の点を持ち、かつ内部が交差しない場合、TRUE を返します。

構文	touches(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `touches( geom_from_wkt( 'LINESTRING(5 3, 4 4)' ), geom_from_wkt( 'LINESTRING(3 3, 4 4, 5 5)' ) )` → TRUE • `touches( geom_from_wkt( 'POINT(4 4)' ), geom_from_wkt( 'POINT(5 5)' ) )` → FALSE

参考： *overlay_touches*

transform

変換元 CRS から変換先 CRS に変換されたジオメトリを返します。

構文	transform(geometry, source_auth_id, dest_auth_id)
引数	• **geometry** - ジオメトリ • **source_auth_id** - 変換元の CRS の ID • **dest_auth_id** - 変換先の CRS の ID
例	• `geom_to_wkt( transform( make_point(488995.53240249, 7104473.38600835), 'EPSG:2154', 'EPSG:4326' ) )` → 'POINT(0 51)'

参考： **レイヤの再投影** アルゴリズム

translate

平行移動させたジオメトリを返します。計算はジオメトリの空間参照系で行われます。

構文	translate(geometry, dx, dy)
引数	• **geometry** - ジオメトリ • **dx** - x 方向の移動量 • **dy** - y 方向の移動量
例	• `translate($geometry, 5, 10)` → 入力ジオメトリと同じタイプのジオメトリ

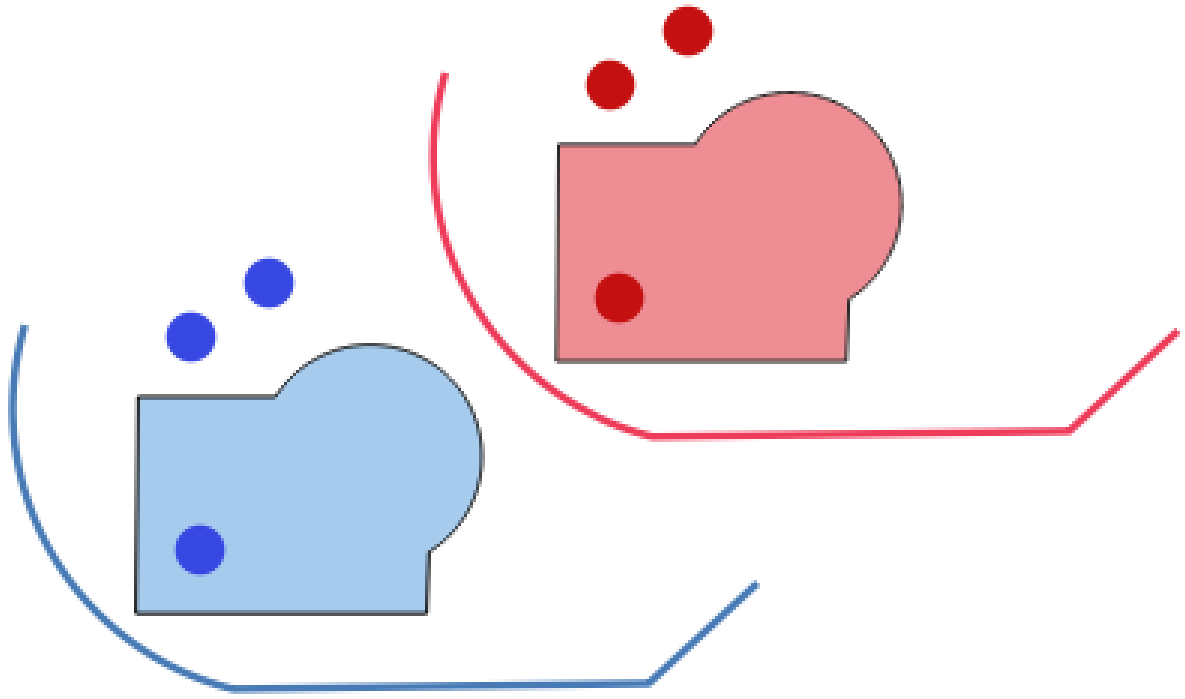


図 13.9: 地物の平行移動

参考： **平行移動（*translate*）** アルゴリズム

triangular_wave

ジオメトリの境界に沿って三角波を構築します。

構文	triangular_wave(geometry, wavelength, amplitude, [strict=False]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **wavelength** - 三角波の波長 • **amplitude** - 三角波の振幅 • **strict** - デフォルトでは、波長の引数は「最大波長」として扱われ、実際の波長はジオメトリの境界に沿って正確な数の三角波が作成されるように動的に調整されます。strict 引数を true に設定すると、波長が正確に使用され、最終的な波形に不完全なパターンが使用される可能性があります。
例	• `triangular_wave(geom_from_wkt('LineString(0 0, 10 0)'), 3, 1)` → ラインストリングに沿った波長 3、振幅 1 の三角波

triangular_wave_randomized

ジオメトリの境界に沿ってランダムな三角波を構築します。

構文	triangular_wave_randomized(geometry, min_wavelength, max_wavelength, min_amplitude, max_amplitude, [seed=0]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **min_wavelength** - 最小の波長 • **max_wavelength** - 最大の波長 • **min_amplitude** - 最小の振幅 • **max_amplitude** - 最大の振幅 • **seed**は、波を生成するための乱数の根を指定します。seed を 0 にすると、完全にランダムな波のセットが生成されます。
例	• `triangular_wave_randomized(geom_from_wkt('LineString(0 0, 10 0)'), 2, 3, 0.1, 0.2)` → ラインストリングに沿った、波長が 2〜3、振幅が 0.1〜0.2 のランダムなサイズの三角波

union

ジオメトリのポイントの和集合を表すジオメトリを返します。

構文	union(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `geom_to_wkt( union( make_point(4, 4), make_point(5, 5) ) )` → 'MULTIPOINT(4 4, 5 5)'

wave

ジオメトリの境界に沿って丸みを帯びた（正弦のような）波を構築します。

構文	wave(geometry, wavelength, amplitude, [strict=False]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **wavelength** - 正弦のような波の波長 • **amplitude** - 正弦のような波の振幅 • **strict** - デフォルトでは、波長の引数は「最大波長」として扱われ、実際の波長はジオメトリの境界に沿って正確な数の波が作成されるように動的に調整されます。strict 引数を true に設定すると、波長が正確に使用され、最終的な波形に不完全なパターンが使用される可能性があります。
例	• `wave(geom_from_wkt('LineString(0 0, 10 0)'), 3, 1)` → ラインストリングに沿った、波長 3、振幅 1 の正弦のような波

wave_randomized

ジオメトリの境界に沿ってランダムな曲線（正弦のような）波を構築します。

構文	wave_randomized(geometry, min_wavelength, max_wavelength, min_amplitude, max_amplitude, [seed=0]) 記号 [] は、オプションの引数を意味します。
引数	• **geometry** - ジオメトリ • **min_wavelength** - 最小の波長 • **max_wavelength** - 最大の波長 • **min_amplitude** - 最小の振幅 • **max_amplitude** - 最大の振幅 • **seed**は、波を生成するための乱数の根を指定します。seed を 0 にすると、完全にランダムな波のセットが生成されます。
例	• `wave_randomized(geom_from_wkt('LineString(0 0, 10 0)'), 2, 3, 0.1, 0.2)` → ラインストリングに沿った、波長が 2〜3、振幅が 0.1〜0.2 のランダムなサイズの丸まった波

wedge_buffer

ポイントジオメトリを起点とするくさび形のバッファを返します。

構文	wedge_buffer(center, azimuth, width, outer_radius, [inner_radius=0.0]) 記号 [] は、オプションの引数を意味します。
引数	• **center** - バッファの中心点（起点）。ポイントジオメトリでなければなりません。 • **azimuth** - くさび形の中央を指す角度（単位は度） • **width** - バッファ幅（度）。くさび形は、方位角の方向にバッファ幅の半分の大きさで両側に伸びることに注意してください。 • **outer_radius** - バッファの外側半径 • **inner_radius** - （オプション）バッファの内側半径
例	• `wedge_buffer(center:=geom_from_wkt('POINT(1 2)'),azimuth:=90, width:=180,outer_radius:=1)` → 向きは東向き、角度の幅は 180 度、外側半径は 1 の、点（1,2）を中心とするくさび形バッファ

参考： **ウェッジバッファを作成** アルゴリズム

within

ジオメトリが別のジオメトリの内部にあるかどうかをテストします。geometry1 が完全に geoemtry2 の内部にある場合、TRUE を返します。

構文	within(geometry1, geometry2)
引数	• **geometry1** - ジオメトリ • **geometry2** - ジオメトリ
例	• `within( geom_from_wkt( 'POINT( 0.5 0.5)' ), geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))' ) )` → TRUE • `within( geom_from_wkt( 'POINT( 5 5 )' ), geom_from_wkt( 'POLYGON((0 0, 0 1, 1 1, 1 0, 0 0 ))' ) )` → FALSE

参考： *overlay_within*

$x

現在のポイント地物の x 座標を返します。マルチポイント地物の場合は、最初のポイントの x 座標を返します。

構文	$x
例	• `$x` → 42

x

ポイントジオメトリの x 座標を返します。ポイントジオメトリではない場合には、重心の x 座標を返します。

構文	x(geometry)
引数	• **geometry** - ジオメトリ
例	• `x( geom_from_wkt( 'POINT(2 5)' ) )` → 2 • `x( $geometry )` → 現在の地物の重心の x 座標

$x_at

現在の地物のジオメトリの x 座標を抜き出します。

構文	$x_at(i)
引数	• **i** - ラインを構成する点のインデックス（インデックスは 0 から始まり、負の値は-1 を最後のインデックスとして逆順で数えます）
例	• `$x_at(1)` → 5

x_max

ジオメトリの x 座標の最大値を返します。計算はこのジオメトリの空間参照系で行われます。

構文	x_max(geometry)
引数	• **geometry** - ジオメトリ
例	• `x_max( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 4

x_min

ジオメトリの x 座標の最小値を返します。計算はこのジオメトリの空間参照系で行われます。

構文	x_min(geometry)
引数	• **geometry** - ジオメトリ
例	• `x_min( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 2

$y

現在のポイント地物の y 座標を返します。マルチポイント地物の場合は、最初のポイントの y 座標を返します。

構文	$y
例	• `$y` → 42

y

ポイントジオメトリの y 座標を返します。ポイントジオメトリではない場合には、重心の y 座標を返します。

構文	y(geometry)
引数	• **geometry** - ジオメトリ
例	• `y( geom_from_wkt( 'POINT(2 5)' ) )` → 5 • `y( $geometry )` → 現在の地物の重心の y 座標

$y_at

現在の地物のジオメトリの y 座標を抜き出します。

構文	$y_at(i)
引数	• **i** - ラインを構成する点のインデックス（インデックスは 0 から始まり、負の値は-1 を最後のインデックスとして逆順で数えます）
例	• `$y_at(1)` → 2

y_max

ジオメトリの y 座標の最大値を返します。計算はこのジオメトリの空間参照系で行われます。

構文	y_max(geometry)
引数	• **geometry** - ジオメトリ
例	• `y_max( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 8

y_min

ジオメトリの y 座標の最小値を返します。計算はこのジオメトリの空間参照系で行われます。

構文	y_min(geometry)
引数	• **geometry** - ジオメトリ
例	• `y_min( geom_from_wkt( 'LINESTRING(2 5, 3 6, 4 8)') )` → 5

$z

地物が 3D の場合に、現在のポイント地物の z 座標を返します。マルチポイント地物の場合は、最初のポイントの z 座標を返します。

構文	$z
例	• `$z` → 123

z

ポイントジオメトリの z 座標を返します。ジオメトリが z 座標値を持たない場合、NULL を返します。

構文	z(geometry)
引数	• **geometry** - ポイントジオメトリ
例	• `z( geom_from_wkt( 'POINTZ(2 5 7)' ) )` → 7

z_max

ジオメトリの z 座標の最大値を返します。z 座標値を持たない場合、NULL を返します。

構文	z_max(geometry)
引数	• **geometry** - z 座標を持つジオメトリ
例	• `z_max( geom_from_wkt( 'POINT ( 0 0 1 )' ) )` → 1 • `z_max( geom_from_wkt( 'MULTIPOINT ( 0 0 1 , 1 1 3 )' ) )` → 3 • `z_max( make_line( make_point( 0,0,0 ), make_point( -1,-1,-2 ) ) )` → 0 • `z_max( geom_from_wkt( 'LINESTRING( 0 0 0, 1 0 2, 1 1 -1 )' ) )` → 2 • `z_max( geom_from_wkt( 'POINT ( 0 0 )' ) )` → NULL

z_min

ジオメトリの z 座標の最小値を返します。z 座標値を持たない場合、NULL を返します。

構文	z_min(geometry)
引数	• **geometry** - z 座標を持つジオメトリ
例	• `z_min( geom_from_wkt( 'POINT ( 0 0 1 )' ) )` → 1 • `z_min( geom_from_wkt( 'MULTIPOINT ( 0 0 1 , 1 1 3 )' ) )` → 1 • `z_min( make_line( make_point( 0,0,0 ), make_point( -1,-1,-2 ) ) )` → -2 • `z_min( geom_from_wkt( 'LINESTRING( 0 0 0, 1 0 2, 1 1 -1 )' ) )` → -1 • `z_min( geom_from_wkt( 'POINT ( 0 0 )' ) )` → NULL

13.2.14 レイアウト関数

このグループには、印刷レイアウトアイテムのプロパティを操作する関数が含まれます。

item_variables

この印刷レイアウト内にあるレイアウトアイテムから、変数をマップ型オブジェクトとして返します。

構文	item_variables(id)
引数	• **id** - レイアウトアイテムの ID
例	• `map_get( item_variables('Map 0'), 'map_scale')` → 現在の印刷レイアウト内にある 'Map 0' アイテムの縮尺

参考：デフォルトの **変数** のリスト

map_credits

レイアウトの地図アイテムに表示されているレイヤのクレジット（使用権）文字列のリストを返します。

構文	map_credits(id, [include_layer_names=false], [layer_name_separator=': ']) 記号 [] は、オプションの引数を意味します。
引数	• **id** - 地図アイテムの ID • **include_layer_names** - true に設定すると、クレジット文字列の前にレイヤ名を含めます • **layer_name_separator** - include_layer_names が true の場合に、レイヤ名とクレジット文字列の間に挿入される文字列
例	• `array_to_string( map_credits( 'Main Map' ) )` → 'Main Map' レイアウトアイテム内に表示されているレイヤのクレジットのコンマ区切りリスト。例： 'CC-BY-NC, CC-BY-SA' • `array_to_string( map_credits( 'Main Map', include_layer_names := true, layer_name_separator := ':  ' ) )` → 'Main Map' レイアウトアイテム内に表示されているレイヤの名前とそのクレジットのコンマ区切りリスト。例： 'Railway lines: CC-BY-NC, Basemap: CC-BY-SA'

この関数は、レイヤの **「アクセス」メタデータプロパティ** が入力されていることを必要とします。

13.2.15 地図レイヤ

このグループには、現在のプロジェクトで利用可能なレイヤーのリストと、各レイヤーのフィールド（データセットに格納されているもの、仮想または補助的なもの、および結合によるもの）が含まれます。フィールドは **フィールドと値** で述べたのと同じように操作することができます。ただし、ダブルクリックすると、フィールドの参照としてではなく、文字列（シングルクォート）として式に追加されます（アクティブレイヤに属さないため）。これは、 *aggregates*、 *attribute*、 *spatial* のクエリを実行するときなど、異なるレイヤーを参照する式を書くのに便利な方法となります。

また、このグループにはレイヤを操作するための便利な関数もあります。

decode_uri

レイヤを引数にとり、元のデータプロバイダの uri をデコードします。どのデータが利用できるかは、データプロバイダに依存します。

構文	decode_uri(layer, [part]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - uri をデコードするレイヤ • **part** - uri の一部。指定しない場合は、uri 全体をマップ型としたものが返ります。
例	• `decode_uri(@layer)` → {'layerId': '0', 'layerName': '', 'path': '/home/qgis/shapefile.shp'} • `decode_uri(@layer)` → {'layerId': NULL, 'layerName': 'layer', 'path': '/home/qgis/geopackage.gpkg'} • `decode_uri(@layer, 'path')` → 'C:\my_data\qgis\shape.shp'

layer_property

該当するレイヤのプロパティまたはメタデータの値を返します。

構文	layer_property(layer, property)
引数	• **layer** - レイヤ名かレイヤ ID のどちらかを表す文字列 • **property** - 戻り値となるプロパティに対応する文字列。有効なオプションは次のとおり： – name: レイヤ名 – id: レイヤ ID – title: メタデータのタイトル文字列 – abstract: メタデータの要約文字列 – keywords: メタデータのキーワード – data_url: メタデータの URL – attribution: メタデータの帰属文字列 – attribution_url: メタデータの帰属 URL – source: レイヤのソース – min_scale: レイヤの最小表示縮尺 – max_scale: レイヤの最大表示縮尺 – is_editable: レイヤが編集モードに入っているか否か – crs: レイヤの CRS – crs_definition: レイヤの CRS の Proj4 定義文字列 – crs_description: レイヤの CRS の名前 – extent: レイヤの範囲（戻り値はジオメトリオブジェクト） – distance_units: レイヤの距離単位 – type: レイヤの種類、すなわち「ベクタ」または「ラスタ」 – storage_type: 保存形式（ベクタレイヤのみ） – geometry_type: ジオメトリの種類。例えば「Point」など（ベクタレイヤのみ） – feature_count: レイヤの地物数の概算値（ベクタレイヤのみ） – path: レイヤのデータソースへのファイルパス。ファイル形式のレイヤのみ利用可
例	• `layer_property('streets','title')` → 'Basemap Streets' • `layer_property('airports','feature_count')` → 120 • `layer_property('landsat','crs')` → 'EPSG:4326'

参考： **ベクタ** 、 **ラスタ** 、 **メッシュ** レイヤのプロパティ

13.2.16 マップ関数

このグループには、キーと値によるマップ型のデータ構造（辞書オブジェクト、キー・バリューのペア、連想配列とも呼ばれます）の作成と操作のための関数が含まれます。値の順序が重要な **リストデータ構造** とは異なり、マップ型オブジェクトにおけるキーと値のペアの順番は重要ではなく、値はそのキーによって識別されます。

from_json

JSON 形式の文字列を読み込みます。

構文	from_json(string)
引数	• **string** - JSON 文字列
例	• `from_json('{"qgis":"rocks"}')` → { 'qgis': 'rocks' } • `from_json('[1,2,3]')` → [1,2,3]

hstore_to_map

hstore 形式の文字列をマップ型オブジェクトに変換します。

構文	hstore_to_map(string)
引数	• **string** - 入力文字列
例	• `hstore_to_map('qgis=>rocks')` → { 'qgis': 'rocks' }

map

パラメータのペアとして渡された、すべてのキーと値のマップ型オブジェクトを返します。

構文	map(key1, value1, key2, value2, ...)
引数	• **key** - キー（文字列） • **value** - 値
例	• `map('1','one','2', 'two')` → { '1': 'one', '2': 'two' } • `map('1','one','2', 'two')['1']` → 'one'

map_akeys

マップ型オブジェクトのすべてのキーを、配列として返します。

構文	map_akeys(map)
引数	• **map** - マップ型オブジェクト
例	• `map_akeys(map('1','one','2','two'))` → ['1', '2']

map_avals

マップ型オブジェクトのすべての値を、配列として返します。

構文	map_avals(map)
引数	• **map** - マップ型オブジェクト
例	• `map_avals(map('1','one','2','two'))` → ['one', 'two']

map_concat

与えられたマップ型の全てのエントリを含むマップ型オブジェクトを返します。2 つのマップ型オブジェクトに同じキーが含まれている場合、値は後者のマップ型のものが採用されます。

構文	map_concat(map1, map2, ...)
引数	• **map** - マップ型オブジェクト
例	• `map_concat(map('1','one', '2','overridden'),map('2','two', '3', 'three'))` → { '1': 'one', '2': 'two', '3': 'three' }

map_delete

指定されたキーとこれに対応する値が削除されたマップ型オブジェクトを返します。

構文	map_delete(map, key)
引数	• **map** - マップ型オブジェクト • **key** - 削除するキー
例	• `map_delete(map('1','one','2','two'),'2')` → { '1': 'one' }

map_exist

指定されたキーがマップ型オブジェクトに存在する場合、TRUE を返します。

構文	map_exist(map, key)
引数	• **map** - マップ型オブジェクト • **key** - 調べたいキー
例	• `map_exist(map('1','one','2','two'),'3')` → FALSE

map_get

マップ型オブジェクトのキーで指定された値を返します。存在しないキーの場合は NULL を返します。

構文	map_get(map, key)
引数	• **map** - マップ型オブジェクト • **key** - 調べたいキー
例	• `map_get(map('1','one','2','two'),'2')` → 'two' • `map_get( item_variables('Map 0'), 'map_scale')` → （存在する場合には）現在の印刷レイアウト内にある 'Map 0' アイテムの縮尺

ヒント: マップ型から値を取得するには **インデックス演算子** *([])* も使用できます。

map_insert

キーと値が追加されたマップ型オブジェクトを返します。キーが既に存在する場合は、その値が上書きされます。

構文	map_insert(map, key, value)
引数	• **map** - マップ型オブジェクト • **key** - 追加するキー • **value** - 追加する値
例	• `map_insert(map('1','one'),'3','three')` → { '1': 'one', '3': 'three' } • `map_insert(map('1','one','2','overridden'),'2','two')` → { '1': 'one', '2': 'two' }

map_prefix_keys

すべてのキーが与えられた文字列でプリフィックスされたマップを返します。

構文	map_prefix_keys(map, prefix)
引数	• **map** - マップ型オブジェクト • **prefix** - 文字列
例	• `map_prefix_keys(map('1','one','2','two'), 'prefix-')` → { 'prefix-1': 'one', 'prefix-2': 'two' }

map_to_hstore

マップ型オブジェクトを hstore 形式の文字列に変換します。

構文	map_to_hstore(map)
引数	• **map** - マップ型オブジェクト
例	• `map_to_hstore(map('qgis','rocks'))` → '"qgis"=>"rocks"'

to_json

マップ型オブジェクトや配列、その他の値などから、JSON 文字列を作成します。

構文	to_json(value)
引数	• **value** - 入力値
例	• `to_json(map('qgis','rocks'))` → {"qgis":"rocks"} • `to_json(array(1,2,3))` → [1,2,3]

url_encode

マップから URL エンコードされた文字列を返します。すべての文字を適切にエンコードした形で変換し、完全準拠のクエリ文字列を生成します。

プラス記号 '+' は変換されないことに注意してください。

構文	url_encode(map)
引数	• **map** - マップ。
例	• `url_encode(map('a&+b', 'a and plus b', 'a=b', 'a equals b'))` → 'a%26+b=a%20and%20plus%20b&a%3Db=a%20equals%20b'

13.2.17 数学関数

このグループには、数学関数（例えば平方根や sin、cos など）が含まれます。

abs

数値の絶対値を返します。

構文	abs(value)
引数	• **value** - 数値
例	• `abs(-2)` → 2

acos

逆余弦（アークコサイン）をラジアンで返します。

構文	acos(value)
引数	• **value** - 角度の余弦
例	• `acos(0.5)` → 1.0471975511966

asin

逆正弦（アークサイン）をラジアンで返します。

構文	asin(value)
引数	• **value** - 角度の正弦
例	• `asin(1.0)` → 1.5707963267949

atan

逆正接（アークタンジェント）をラジアンで返します。

構文	atan(value)
引数	• **value** - 角度の正接
例	• `atan(0.5)` → 0.463647609000806

atan2

2 つの引数の符号を用いて象限を決定し、dy/dx の逆正接（アークタンジェント）を返します。

構文	atan2(dy, dx)
引数	• **dy** - y 座標の差 • **dx** - x 座標の差
例	• `atan2(1.0, 1.732)` → 0.523611477769969

ceil

数値を上に丸めた値を返します。

構文	ceil(value)
引数	• **value** - 数値
例	• `ceil(4.9)` → 5 • `ceil(-4.9)` → -4

clamp

入力値を指定の範囲に制限した値を返します。

構文	clamp(minimum, input, maximum)
引数	• **minimum** - *input* が取りうる最小値 • **input** - *minimum* と *maximum* で指定される範囲に制限される値 • **maximum** - *input* が取りうる最大値
例	• `clamp(1,5,10)` → 5 *input* は 1 と 10 の間にあるので、そのまま返します。 • `clamp(1,0,10)` → 1 *input* は最小値の 1 より小さいので、関数は 1 を返します。 • `clamp(1,11,10)` → 10 *input* は最大値の 10 より大きいので、関数は 10 を返します。

cos

角度の余弦（コサイン）を返します。

構文	cos(angle)
引数	• **angle** - ラジアン単位の角度
例	• `cos(1.571)` → 0.000796326710733263

degrees

ラジアンから度に変換します。

構文	degrees(radians)
引数	• **radians** - 数値
例	• `degrees(3.14159)` → 180 • `degrees(1)` → 57.2958

exp

自然対数の底（ネイピア数）のべき乗を返します。

構文	exp(value)
引数	• **value** - 指数
例	• `exp(1.0)` → 2.71828182845905

floor

数値を下に丸めた値を返します。

構文	floor(value)
引数	• **value** - 数値
例	• `floor(4.9)` → 4 • `floor(-4.9)` → -5

ln

自然対数を返します。

構文	ln(value)
引数	• **value** - 数値
例	• `ln(1)` → 0 • `ln(2.7182818284590452354)` → 1

log

指定した底の対数を返します。

構文	log(base, value)
引数	• **base** - 正の数 • **value** - 正の数
例	• `log(2, 32)` → 5 • `log(0.5, 32)` → -5

log10

10 を底とする、常用対数を返します。

構文	log10(value)
引数	• **value** - 正の数
例	• `log10(1)` → 0 • `log10(100)` → 2

max

最大値を返します。

構文	max(value1, value2, ...)
引数	• **value** - 数値
例	• `max(2,10.2,5.5)` → 10.2 • `max(20.5,NULL,6.2)` → 20.5

min

最小値を返します。

構文	min(value1, value2, ...)
引数	• **value** - 数値
例	• `min(20.5,10,6.2)` → 6.2 • `min(2,-10.3,NULL)` → -10.3

pi

円周率を返します。

構文	pi()
例	• `pi()` → 3.14159265358979

radians

角度をラジアンに変換します。

構文	radians(degrees)
引数	• **degrees** - 数値
例	• `radians(180)` → 3.14159 • `radians(57.2958)` → 1

rand

最小値と最大値の引数で指定された範囲内（境界値を含む）の、ランダムな整数を返します。乱数のシードを指定する場合、値はシードに応じて常に同じ値となります。

構文	rand(min, max, [seed=NULL]) 記号 [] は、オプションの引数を意味します。
引数	• **min** - 乱数の下限値の整数 • **max** - 乱数の上限値の整数 • **seed** - 乱数のシードとして使用する任意の値
例	• `rand(1, 10)` → 8

randf

最小値と最大値の引数で指定された範囲内（境界値を含む）の、ランダムな浮動小数点数を返します。乱数のシードを指定する場合、値はシードに応じて常に同じ値となります。

構文	randf([min=0.0], [max=1.0], [seed=NULL]) 記号 [] は、オプションの引数を意味します。
引数	• **min** - 乱数の下限値の浮動小数点数 • **max** - 乱数の上限値の浮動小数点数 • **seed** - 乱数のシードとして使用する任意の値
例	• `randf(1, 10)` → 4.59258286403147

round

指定した小数点以下の桁数になるように四捨五入した数値を返します。

構文	round(value, [places=0]) 記号 [] は、オプションの引数を意味します。
引数	• **value** - 丸められる数値 • **places** - （オプション）数値を丸める桁を表す整数。負の値も可。
例	• `round(1234.567, 2)` → 1234.57 • `round(1234.567)` → 1235 • `round(1234.567, -1)` → 1230

scale_exp

指数曲線を用いて、与えられた値を入力範囲から出力範囲に変換します。この関数は、指定された出力範囲へ値をイージングするために使用します。

構文	scale_exp(value, domain_min, domain_max, range_min, range_max, exponent)
引数	• **value** - 入力範囲内の値。関数は出力範囲でスケーリングされた対応する値を返します。 • **domain_min** - 入力範囲の最小値を指定します。入力値が取りうる最小値です。 • **domain_max** - 入力範囲の最大値を指定します。入力値が取りうる最大値です。 • **range_min** - 出力範囲の最小値を指定します。関数による出力の最小値です。 • **range_max** - 出力範囲の最大値を指定します。関数による出力の最大値です。 • **exponent** - 入力値を出力範囲へマッピングする方法を決定する正の値（0 より大きな値）。指数が大きいと出力値は 'ease in' され、出力値の増加はゆっくりと始まって、入力値が入力範囲の最大値に近づくにつれて増加が加速します。指数が小さい（1 よりも小さい）と出力値は 'ease out' され、値の増加は早く始まり、入力範囲の最大値に近づくにつれて増加は減速します。
例	• `scale_exp(5,0,10,0,100,2)` → 25 指数 2 を使用した easing in • `scale_exp(3,0,10,0,100,0.5)` → 54.772 指数 0.5 を使用した easing out

scale_linear

線形補間を用いて、与えられた値を入力範囲から出力範囲に変換します。

構文	scale_linear(value, domain_min, domain_max, range_min, range_max)
引数	• **value** - 入力範囲内の値。関数は出力範囲でスケーリングされた対応する値を返します。 • **domain_min** - 入力範囲の最小値を指定します。入力値が取りうる最小値です。 • **domain_max** - 入力範囲の最大値を指定します。入力値が取りうる最大値です。 • **range_min** - 出力範囲の最小値を指定します。関数による出力の最小値です。 • **range_max** - 出力範囲の最大値を指定します。関数による出力の最大値です。
例	• `scale_linear(5,0,10,0,100)` → 50 • `scale_linear(0.2,0,1,0,360)` → 72 0 から 1 までの値を、0 から 360 までの角度に変換します • `scale_linear(1500,1000,10000,9,20)` → 9.6111111 1000 から 10000 まで変化する人口を 9 から 20 のフォントサイズにスケーリングします。

sin

角度の正弦（サイン）を返します。

構文	sin(angle)
引数	• **angle** - ラジアン単位の角度
例	• `sin(1.571)` → 0.999999682931835

sqrt

平方根を返します。

構文	sqrt(value)
引数	• **value** - 数値
例	• `sqrt(9)` → 3

tan

角度の正接（タンジェント）を返します。

構文	tan(angle)
引数	• **angle** - ラジアン単位の角度
例	• `tan(1.0)` → 1.5574077246549

13.2.18 メッシュ関数

このグループには、メッシュに関連した値を計算したり返したりする関数が含まれます。

$face_area

現在のメッシュ面の面積を返します。この関数で計算される面積には、現在のプロジェクトの楕円体設定と面積単位設定の両方が反映されます。例えば、プロジェクトに回転楕円体が設定されている場合、楕円体面積になり、設定されていない場合、平面上の面積になります。

構文	$face_area
例	• `$face_area` → 42

$face_index

現在のメッシュ面のインデックスを返します。

構文	$face_index
例	• `$face_index` → 4581

$vertex_as_point

現在の頂点をポイントジオメトリとして返します。

構文	$vertex_as_point
例	• `geom_to_wkt( $vertex_as_point )` → 'POINT(800 1500 41)'

$vertex_index

現在のメッシュ頂点のインデックスを返します。

構文	$vertex_index
例	• `$vertex_index` → 9874

$vertex_x

現在のメッシュ頂点の x 座標を返します。

構文	$vertex_x
例	• `$vertex_x` → 42.12

$vertex_y

現在のメッシュ頂点の y 座標を返します。

構文	$vertex_y
例	• `$vertex_y` → 12.24

$vertex_z

現在のメッシュ頂点の z 座標を返します。

構文	$vertex_z
例	• `$vertex_z` → 42

13.2.19 演算子

このグループには演算子が含まれています（例えば、+、 - 、*）。以下の数学関数のほとんどについて、入力の一つが NULL である場合は結果は NULL であることに注意してください。

%

除算の余り。被除数の符号をとる。

構文	a % b
引数	• **a** - 値 • **b** - 値
例	• `9 % 2` → 1 • `9 % -2` → 1 • `-9 % 2` → -1 • `5 % NULL` → NULL

*

2 つの値の乗算

構文	a * b
引数	• **a** - 値 • **b** - 値
例	• `5 * 4` → 20 • `5 * NULL` → NULL

+

2 つの値の加算。いずれかの値が NULL の場合、結果は NULL になります。

構文	a + b
引数	• **a** - 値 • **b** - 値
例	• `5 + 4` → 9 • `5 + NULL` → NULL • `'QGIS ' + 'ROCKS'` → 'QGIS ROCKS' • `to_datetime('2020-08-01 12:00:00') + '1 day 2 hours'` → 2020-08-02T14:00:00

参考： *concat* 、 ||

-

2 つの値の減算。いずれかの値が NULL の場合、結果は NULL になります。

構文	a - b
引数	• **a** - 値 • **b** - 値
例	• `5 - 4` → 1 • `5 - NULL` → NULL • `to_datetime('2012-05-05 12:00:00') - to_interval('1 day 2 hours')` → 2012-05-04T10:00:00

/

2 つの値の除算

構文	a / b
引数	• **a** - 値 • **b** - 値
例	• 5 / 4 → 1.25 • 5 / NULL → NULL

<

2 つの値を比較し、左の値が右の値より小さい場合、1 と評価されます。

構文	a < b
引数	• **a** - 値 • **b** - 値
例	• 5 < 4 → FALSE • 5 < 5 → FALSE • 4 < 5 → TRUE

<=

2 つの値を比較し、左の値が右の値より小さいか等しい場合、1 と評価されます。

構文	a <= b
引数	• **a** - 値 • **b** - 値
例	• 5 <= 4 → FALSE • 5 <= 5 → TRUE • 4 <= 5 → TRUE

<>

2 つの値を比較し、両者が等しくない場合、1 と評価されます。

構文	a <> b
引数	• **a** - 値 • **b** - 値
例	• 5 <> 4 → TRUE • 4 <> 4 → FALSE • 5 <> NULL → NULL • NULL <> NULL → NULL

=

2 つの値を比較し、両者が等しい場合、1 と評価されます。

構文	a = b
引数	• **a** - 値 • **b** - 値
例	• 5 = 4 → FALSE • 4 = 4 → TRUE • 5 = NULL → NULL • NULL = NULL → NULL

>

2 つの値を比較し、左の値が右の値より大きい場合、1 と評価されます。

構文	a > b
引数	• **a** - 値 • **b** - 値
例	• 5 > 4 → TRUE • 5 > 5 → FALSE • 4 > 5 → FALSE

>=

2 つの値を比較し、左の値が右の値より大きいか等しい場合、1 と評価されます。

構文	a >= b
引数	• **a** - 値 • **b** - 値
例	• 5 >= 4 → TRUE • 5 >= 5 → TRUE • 4 >= 5 → FALSE

AND

条件式 a と条件式 b がともに true である場合に TRUE を返します。

構文	a AND b
引数	• **a** - 条件式 • **b** - 条件式
例	• `TRUE AND TRUE` → TRUE • `TRUE AND FALSE` → FALSE • `4 = 2+2 AND 1 = 1` → TRUE • `4 = 2+2 AND 1 = 2` → FALSE

BETWEEN

値が指定された範囲内にあるとき、TRUE を返します。範囲は、境界を含むとみなされます。除外を調べるには、NOT BETWEEN を使うことができます。

構文	value BETWEEN lower_bound AND higher_bound
引数	• **value** - 範囲と比較する値。文字列、数字または日付です。 • **lower_bound AND higher_bound** - 範囲
例	• `'B' BETWEEN 'A' AND 'C'` → TRUE • `2 BETWEEN 1 AND 3` → TRUE • `2 BETWEEN 2 AND 3` → TRUE • `'B' BETWEEN 'a' AND 'c'` → FALSE • `lower('B') BETWEEN 'a' AND 'b'` → TRUE

注釈: *value BETWEEN lower_bound AND higher_bound* は "*value >= lower_bound AND value <= higher_bound*" と同じです。

参考： *NOT BETWEEN*

ILIKE

最初の引数が与えられたパターンと大文字・小文字を区別せずに一致する場合に TRUE を返します。マッチングで大文字・小文字を区別する場合には、ILIKE の代わりに LIKE を使用してください。数値にも対応しています。

構文	string/number ILIKE pattern
引数	• **string/number** - 検索する文字列 • **pattern** - 検索するパターン。 '%' をワイルドカードとして、 '_' を任意の 1 文字として、 '\\' をこれらの特殊文字のエスケープとして利用できます。
例	• `'A' ILIKE 'A'` → TRUE • `'A' ILIKE 'a'` → TRUE • `'A' ILIKE 'B'` → FALSE • `'ABC' ILIKE 'b'` → FALSE • `'ABC' ILIKE 'B'` → FALSE • `'ABC' ILIKE '_b_'` → TRUE • `'ABC' ILIKE '_B_'` → TRUE • `'ABCD' ILIKE '_b_'` → FALSE • `'ABCD' ILIKE '_B_'` → FALSE • `'ABCD' ILIKE '_b%'` → TRUE • `'ABCD' ILIKE '_B%'` → TRUE • `'ABCD' ILIKE '%b%'` → TRUE • `'ABCD' ILIKE '%B%'` → TRUE • `'ABCD%' ILIKE 'abcd\\%'` → TRUE • `'ABCD' ILIKE '%B\\%'` → FALSE

IN

リストの中に値が見つかった場合に TRUE を返します。

構文	a IN b
引数	• **a** - 値 • **b** - 値のリスト
例	• `'A' IN ('A','B')` → TRUE • `'A' IN ('C','B')` → FALSE

IS

a と b が同じ場合に TRUE を返します。

構文	a IS b
引数	• **a** - 任意の値 • **b** - 任意の値
例	• `'A' IS 'A'` → TRUE • `'A' IS 'a'` → FALSE • `4 IS 4` → TRUE • `4 IS 2+2` → TRUE • `4 IS 2` → FALSE • `$geometry IS NULL` → ジオメトリが NULL でない場合は 0

IS NOT

a と b が同じではない場合に TRUE を返します。

構文	a IS NOT b
引数	• **a** - 値 • **b** - 値
例	• `'a' IS NOT 'b'` → TRUE • `'a' IS NOT 'a'` → FALSE • `4 IS NOT 2+2` → FALSE

LIKE

最初の引数が与えられたパターンと一致する場合に TRUE を返します。数値にも対応しています。

構文	string/number LIKE pattern
引数	• **string/number** - 値 • **pattern** - 値と比較するパターン。 '%' をワイルドカードとして、 '_' を任意の 1 文字として、 '\\' をこれらの特殊文字のエスケープとして利用できます。
例	• `'A' LIKE 'A'` → TRUE • `'A' LIKE 'a'` → FALSE • `'A' LIKE 'B'` → FALSE • `'ABC' LIKE 'B'` → FALSE • `'ABC' LIKE '_B_'` → TRUE • `'ABCD' LIKE '_B_'` → FALSE • `'ABCD' LIKE '_B%'` → TRUE • `'ABCD' LIKE '%B%'` → TRUE • `'1%' LIKE '1\\%'` → TRUE • `'1_' LIKE '1\\%'` → FALSE

NOT

条件式の否定を返します。

構文	NOT a
引数	• **a** - 条件式
例	• `NOT 1` → FALSE • `NOT 0` → TRUE

NOT BETWEEN

値が指定された範囲内にない場合、TRUE を返します。範囲は、境界を含むとみなされます。

構文	value NOT BETWEEN lower_bound AND higher_bound
引数	• **value** - 範囲と比較する値。文字列、数字または日付です。 • **lower_bound AND higher_bound** - 範囲
例	• `'B' NOT BETWEEN 'A' AND 'C'` → FALSE • `1.0 NOT BETWEEN 1.1 AND 1.2` → TRUE • `2 NOT BETWEEN 2 AND 3` → FALSE • `'B' NOT BETWEEN 'a' AND 'c'` → TRUE • `lower('B') NOT BETWEEN 'a' AND 'b'` → FALSE

注釈: *value NOT BETWEEN lower_bound AND higher_bound* は "*value < lower_bound OR value > higher_bound*" と同じです。

参考： *BETWEEN*

OR

条件式 a、条件式 b のいずれかが true である場合に TRUE を返します。

構文	a OR b
引数	• **a** - 条件式 • **b** - 条件式
例	• `4 = 2+2 OR 1 = 1` → TRUE • `4 = 2+2 OR 1 = 2` → TRUE • `4 = 2 OR 1 = 2` → FALSE

[]

インデックス演算子です。配列の要素やマップ型の値を返します。

構文	[index]
引数	• **index** - 配列のインデックスまたはマップ型のキー
例	• `array(1,2,3)[0]` → 1 • `array(1,2,3)[2]` → 3 • `array(1,2,3)[-1]` → 3 • `map('a',1,'b',2)['a']` → 1 • `map('a',1,'b',2)['b']` → 2

参考： *array_get* 、 *map_get*

^

2 つの値の累乗。

構文	a ^ b
引数	• **a** - 値 • **b** - 値
例	• `5 ^ 4` → 625 • `5 ^ NULL` → NULL

||

2 つの値を連結して文字列にします。

いずれかの値が NULL の場合、結果は NULL になります。動作が異なる concat 関数についても参照してください。

構文	a || b
引数	• **a** - 値 • **b** - 値
例	• `'Here' || ' and ' || 'there'` → 'Here and there' • `'Nothing' || NULL` → NULL • `'Dia: ' || "Diameter"` → 'Dia: 25' • `1 || 2` → '12'

参考： *concat* 、 *+*

~

文字列に対して正規表現のマッチングを実行します。バックスラッシュ文字は二重にエスケープする必要があります（例：空白文字に一致させるには "\\s" ）

構文	string ~ regex
引数	• **string** - 文字列 • **regex** - 正規表現。バックスラッシュにはエスケープが必要です。例： \\d.
例	• `'hello' ~ 'll'` → TRUE • `'hello' ~ '^ll'` → FALSE • `'hello' ~ 'llo$'` → TRUE • `'abc123' ~ '\\d+'` → TRUE

参考： *regexp_match*

13.2.20 プロセシング関数

このグループには、プロセシングアルゴリズムで使用できる関数が含まれます。

- *parameter*

parameter

プロセシングアルゴリズムの入力パラメータの値を返します。

構文	parameter(name)
引数	• **name** - 対応する入力パラメータの名前
例	• `parameter('BUFFER_SIZE')` → 5.6

13.2.21 ラスタ関数

このグループには、ラスタレイヤを操作するための関数が含まれます。

raster_statistic

ラスタレイヤの統計値を返します。

構文	raster_statistic(layer, band, property)
引数	• **layer** - ラスタレイヤ名またはレイヤ ID を表す文字列 • **band** - 1 から始まる、ラスタレイヤのバンド番号を表す整数 • **property** - 戻り値となるプロパティに対応する文字列。有効なオプションは次のとおり： – min: 最小値 – max: 最大値 – avg: 平均値 – stdev: 標準偏差 – range: 値の範囲（max - min） – sum: ラスタの値の合計値
例	• `raster_statistic('lc',1,'avg')` → 'lc' ラスタレイヤのバンド 1 の平均値 • `raster_statistic('ac2010',3,'min')` → 'ac2010' ラスタレイヤのバンド 3 の最小値

raster_value

指定された点のラスタ値を返します。

構文	raster_value(layer, band, point)
引数	• **layer** - ラスタレイヤの名前または id • **band** - 値をサンプリングするバンド番号 • **point** - ポイントジオメトリ（2 つ以上のパートを持つマルチパートジオメトリの場合には、NULL を返します）
例	• `raster_value('dem', 1, make_point(1,1))` → 25

13.2.22 レコードと属性関数

このグループには、レコード識別子を操作する関数が含まれます。

attribute

地物の属性を返します。

バージョン 1

現在の地物の属性値を返します。

構文	attribute(attribute_name)
引数	• **attribute_name** - 返される属性名
例	• `attribute( 'name' )` → 現在の地物の 'name' 属性に格納されている値

バージョン 2

指定された地物の属性値を返します。

構文	attribute(feature, attribute_name)
引数	• **feature** - 地物 • **attribute_name** - 返される属性名
例	• `attribute( @atlas_feature, 'name' )` → 現在の地図帳地物の 'name' 属性に格納されている値

attributes

属性名をキーとする、地物のすべての属性を含むマップ型オブジェクトを返します。

バージョン 1

現在の地物の全属性をマップ型オブジェクトとして返します。

構文	attributes()
例	• `attributes()['name']` → 現在の地物の 'name' 属性に格納されている値

バージョン 2

ターゲットとする地物を指定できます。

構文	attributes(feature)
引数	• **feature** - 地物
例	• `attributes( @atlas_feature )['name']` → 現在の地図帳地物の 'name' 属性に格納されている値

参考： **マップ関数**

$currentfeature

現在評価されている地物を返します。'attribute' 関数と併用することで、現在の地物から属性値を評価することができます。**警告: この関数は非推奨です。代わりに代替の@feature 変数を使用することが推奨されます。**

構文	$currentfeature
例	• `attribute( $currentfeature, 'name' )` → 現在の地物の 'name' 属性に格納されている値

display_expression

指定された地物の表示式を返します。式はデフォルトで評価されます。0 個、1 個、または 2 個以上の引数をとることができ、詳しくは以下の通りです。

引数なしバージョン

引数なしで関数を呼んだ場合、関数は現在のレイヤの現在の地物の表示式を評価します。

構文	display_expression()
例	• `display_expression()` → 現在のレイヤの現在の地物の表示式

引数が地物 1 つのバージョン

地物だけを引数として関数を呼んだ場合、関数は現在のレイヤの指定された地物の表示式を評価します。

構文	display_expression(feature)
引数	• **feature** - 表示式を評価したい地物
例	• `display_expression(@atlas_feature)` → 現在の地図帳地物の表示式

レイヤと地物の引数バージョン

レイヤと地物の両方を引数にとって関数を呼んだ場合、関数は指定されたレイヤの指定された地物の表示式を評価します。

構文	display_expression(layer, feature, [evaluate=true]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - レイヤ（ID または名前） • **feature** - 表示式を評価したい地物 • **evaluate** - 表示式が評価される必要があるかどうかを指定します。false とすると、表示式は文字列リテラルで返されるだけです（後で 'eval' 関数を使用して評価する必要があります）
例	• `display_expression( 'streets', get_feature_by_id('streets', 1))` → 'streets' レイヤの、ID が 1 の地物の表示式 • `display_expression('a_layer_id', $currentfeature, 'False')` → 指定された地物の評価前の表示式

get_feature

指定された属性値にマッチした最初の地物を返します。

単独の値のバージョン

レイヤ ID、および、ひとつの列と値を指定します。

構文	get_feature(layer, attribute, value)
引数	• **layer** - レイヤ名または ID • **attribute** - マッチに使う属性名 • **value** - マッチさせたい属性値
例	• `get_feature('streets','name','main st')` → "name" フィールドの値が "main st" となっている、 "streets" レイヤで最初に見つかった地物

マップ型バージョン

レイヤ ID、および、使用する列（キー）とその値を含むマップ

構文	get_feature(layer, attribute)
引数	• **layer** - レイヤ名または ID • **attribute** - 使用する列と値のペアを含んでいるマップ
例	• `get_feature('streets',map('name','main st','lane_num','4'))` → "name"フィールドの値が"main st"、"lane_num"フィールドの値が"4"である、"streets"レイヤの最初の地物

get_feature_by_id

ID で指定した地物を返します。

構文	get_feature_by_id(layer, feature_id)
引数	• **layer** - レイヤ名またはレイヤ ID • **feature_id** - 戻り値として取得したい地物の ID
例	• `get_feature_by_id('streets', 1)` → "streets" レイヤの ID が 1 の地物

参考： *$id*

$id

現在の行の地物 ID を返します。**警告: この関数は非推奨です。代わりに代替の@id 変数を使用することをお勧めします。**

構文	$id
例	• `$id` → 42

is_selected

地物が選択されている場合に TRUE を返します。0 個、1 個、または 2 個の引数をとることができ、詳しくは以下の通りです。

引数なしバージョン

引数なしで関数を呼んだ場合、関数は現在のレイヤの現在の地物が選択されている場合に TRUE を返します。

構文	is_selected()
例	• `is_selected()` → 現在のレイヤの現在の地物が選択されている場合、TRUE

引数が地物 1 つのバージョン

'feature' だけを引数として関数を呼んだ場合、関数は現在のレイヤの指定された地物が選択されている場合に TRUE を返します。

構文	is_selected(feature)
引数	• **feature** - 選択されているかどうかを確認したい地物
例	• `is_selected(@atlas_feature)` → 現在の地図帳地物が選択されている場合、TRUE。 • `is_selected(get_feature('streets', 'name', 'Main St.'))` → アクティブな "streets" レイヤで地物に一意に名前が付けられ、その中で "Main St." 地物が選択されている場合に TRUE。 • `is_selected(get_feature_by_id('streets', 1))` → アクティブな "streets" レイヤで id=1 の地物が選択されている場合に TRUE

引数が 2 つのバージョン

レイヤと地物の両方を引数にとって関数を呼んだ場合、関数は指定されたレイヤの指定された地物が選択されている場合に TRUE を返します。

構文	is_selected(layer, feature)
引数	• **layer** - 地物の選択をチェックしたいレイヤ（ID またはレイヤ名） • **feature** - 選択されているかどうかを確認したい地物
例	• `is_selected( 'streets', get_feature('streets', 'name', "street_name"))` → 現在の建物の street が選択されていれば TRUE（建物レイヤに'street_name' というフィールドがあり、'streets' レイヤにユニークな値を持つ'name' というフィールドがあると仮定している）。 • `is_selected( 'streets', get_feature_by_id('streets', 1))` → "streets" レイヤの id 1 の地物が選択されている場合に TRUE。

maptip

指定された地物の Tip を返します。式はデフォルトで評価されます。0 個、1 個、または 2 個以上の引数をとることができ、詳しくは以下の通りです。

引数なしバージョン

引数なしで関数を呼んだ場合、関数は現在のレイヤの現在の地物の Tip を評価します。

構文	maptip()
例	• `maptip()` → 現在のレイヤの現在の地物の Tip

引数が地物 1 つのバージョン

地物だけを引数として関数を呼んだ場合、関数は現在のレイヤの指定された地物の表示式を評価します。

構文	maptip(feature)
引数	• **feature** - 表示式を評価したい地物
例	• `maptip(@atlas_feature)` → 現在の地図帳地物の Tip

レイヤと地物の引数バージョン

レイヤと地物の両方を引数にとって関数を呼んだ場合、関数は指定されたレイヤの指定された地物の表示式を評価します。

構文	maptip(layer, feature, [evaluate=true]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - レイヤ（ID または名前） • **feature** - 表示式を評価したい地物 • **evaluate** - 地物の Tip が評価される必要があるかどうかを指定します。false とすると、表示式は文字列リテラルで返されるだけです（後で 'eval_template' 関数を使用して評価する必要があります）
例	• `maptip('streets', get_feature_by_id('streets', 1))` → 'streets' レイヤの、ID が 1 の地物の Tip • `maptip('a_layer_id', $currentfeature, 'False')` → 指定された地物の評価前の Tip

num_selected

指定されたレイヤで選択されている地物の数を返します。デフォルトでは、式が評価されるレイヤ上で動作します。

構文	num_selected([layer=current layer]) 記号 [] は、オプションの引数を意味します。
引数	• **layer** - 地物の選択をチェックしたいレイヤ（ID またはレイヤ名）
例	• `num_selected()` → 現在のレイヤで選択されている地物の数 • `num_selected('streets')` → 'streets' レイヤ上で選択されている地物の数

represent_attributes

属性名をキーとし、設定された表現値を値とするマップを返します。属性の表現値は、各属性に設定されているウィジェットタイプに依存します。 0 個、1 個、またはそれ以上の引数とともに使用することができます。詳細は以下をご覧ください。

引数なしバージョン

パラメータなしで呼び出された場合、この関数は、現在のレイヤの地物の属性の表現を返します。

構文	represent_attributes()
例	• `represent_attributes()` → 現在の地物の属性の表現。

引数が地物 1 つのバージョン

'feature' パラメータだけで呼び出した場合、関数は現在のレイヤから指定された地物の属性の表現を返します。

構文	represent_attributes(feature)
引数	• **feature** - 表示式を評価したい地物
例	• `represent_attributes(@atlas_feature)` → 現在のレイヤから指定された地物の属性の表現。

レイヤと地物の引数バージョン

'layer' と 'feature' パラメータで呼び出した場合、関数は指定されたレイヤから指定された地物の属性の表現を返します。

構文	represent_attributes(layer, feature)
引数	• **layer** - レイヤ（ID またはレイヤ名）。 • **feature** - 表示式を評価したい地物
例	• `represent_attributes('atlas_layer', @atlas_feature)` → 指定されたレイヤから指定された地物の属性の表現。

参考： *represent_value*

represent_value

フィールド値に設定された表現値を返します。結果はウィジェットのタイプによって異なります。多くの場合、この関数は「バリューマップ」ウィジェットで便利です。

構文	represent_value(value, [fieldName]) 記号 [] は、オプションの引数を意味します。
引数	• **value** - 評価する値。ほとんどの場合はフィールドを指定。 • **fieldName** - ウィジェットの設置が読み込まれるフィールド名。
例	• `represent_value("field_with_value_map")` → バリューマップの値の説明 • `represent_value('static value', 'field_name')` → 'field_name' が static value であることの説明

参考： *widget types*, *represent_attributes*

sqlite_fetch_and_increment

SQLite データベースの自動インクリメントを管理します。

SQLite のデフォルト値は insert 時にしか適用されず、先読みすることはできません。

このため、データベースに行を作成する前に AUTO_INCREMENT でインクリメントされた主キーを取得することはできません。補足：postgres では、 **デフォルト値を評価する** オプションでこれが可能です。

リレーションを持つ新たな地物を追加する際、親地物のフォームがまだ開いていてコミットされていない状態でも、親に子を追加することができると便利です。

この制限を回避するため、gpkg のような sqlite ベースの形式で、この関数を使用してシーケンス値を別のテーブルで管理できます。

シーケンステーブルはシーケンス ID に対してフィルタリングされ（filter_attribute と filter_value）、id_field の現在の値に 1 増やした値が返されます。

別のカラムで値を指定する必要がある場合には、default_values マップを使用できます。

注意

この関数はターゲットの SQLite テーブルを更新します。これは、各属性のデフォルト値設定を使用することが想定されているためです。

データベースのパラメータがレイヤで、レイヤがトランザクションモードの場合、値はトランザクション期間中に一度だけ取得され、キャッシュされて値がインクリメントされます。このため、同じデータベース上で複数のプロセスから並列に作業することは安全ではありません。

構文	sqlite_fetch_and_increment(database, table, id_field, filter_attribute, filter_value, [default_values]) 記号 [] は、オプションの引数を意味します。
引数	• **database** - SQLite ファイルまたは GeoPackage レイヤへのパス • **table** - シーケンスを管理するテーブルの名前 • **id_field** - 現在の id 値を含んでいるフィールドの名前 • **filter_attribute** - このシーケンスの一意の識別子を含むフィールドの名前。UNIQUE インデックスが必要です。 • **filter_value** - シーケンスに使用する名前 • **default_values** - テーブルの別カラムのデフォルト値のためのマップオブジェクト。値は全体をクォート記号で括る必要があります。関数も可能です。
例	• `sqlite_fetch_and_increment(@layer, 'sequence_table', 'last_unique_id', 'sequence_id', 'global', map('last_change', 'date(''now'')', 'user', '''' || @user_account_name || ''''))` → 0 • `sqlite_fetch_and_increment(layer_property(@layer, 'path'), 'sequence_table', 'last_unique_id', 'sequence_id', 'global', map('last_change', 'date(''now'')', 'user', '''' || @user_account_name || ''''))` → 0

参考： **データソースプロパティ** 、 *1* **対多または多対多のリレーションの作成**

uuid

Qt の QUuid::createUuid メソッドを使用して、テーブルの各行の UUID（Universally Unique Identifier）を生成します。

構文	uuid([format='WithBraces']) 記号 [] は、オプションの引数を意味します。
引数	• **format** - UUID のフォーマット。 'WithBraces' 、 'WithoutBraces' 、または 'Id128'
例	• `uuid()` → '{0bd2f60f-f157-4a6d-96af-d4ba4cb366a1}' • `uuid('WithoutBraces')` → '0bd2f60f-f157-4a6d-96af-d4ba4cb366a1' • `uuid('Id128')` → '0bd2f60ff1574a6d96afd4ba4cb366a1'

13.2.23 リレーション

このグループには、このプロジェクトで利用可能な **リレーション** のリストとその説明が含まれています。このグループは（例えば *relation_aggregate* 関数を使って）式を書く際やフォームをカスタマイズする際に、リレーション ID に素早くアクセスすることができます。

13.2.24 文字列関数

このグループには、文字列を操作する関数が含まれます（置換、大文字に変換など）。

ascii

文字列の最初の文字の unicode 番号を返します。

構文	ascii(string)
引数	• **string** - unicode 番号に変換する文字列
例	• `ascii('Q')` → 81

char

unicode 番号に対応した文字を返します。

構文	char(code)
引数	• **code** - unicode 番号
例	• `char(81)` → 'Q'

concat

複数の文字列を 1 つに結合したものを返します。NULL 値は空文字列に変換されます。その他の値（数値など）は文字列に変換されます。

構文	concat(string1, string2, ...)
引数	• **string** - 文字列
例	• `concat('sun', 'set')` → 'sunset' • `concat('a','b','c','d','e')` → 'abcde' • `concat('Anno ', 1984)` → 'Anno 1984' • `concat('The Wall', NULL)` → 'The Wall'

フィールド連結について

文字列やフィールド値の連結には || や + 演算子を使用することもできますが、これらにはいくつか特性があります。

- + 演算子は式の合計も意味するので、整数（フィールドや数値）オペランドがある場合にはエラーを起こしやすく、他の演算子を使用する方が良いでしょう:

```
'My feature id is: ' + "gid" => triggers an error as gid returns an integer
```

- 引数のどれかが NULL 値の場合、 || や + は NULL を返します。NULL 値の有無に関わらず、その他の引数の結合を返したい場合には、 concat 関数を使用するのが良いでしょう:

```
'My feature id is: ' + NULL ==> NULL
'My feature id is: ' || NULL => NULL
concat('My feature id is: ', NULL) => 'My feature id is: '
```

参考： || 、 +

format

指定された引数を用いて文字列をフォーマットします。

構文	format(string, arg1, arg2, ...)
引数	• **string** - 引数のプレースホルダを含む文字列。プレースホルダには、 %1 、 %2 などを使います。プレースホルダは繰り返し使用できます。最小番号のプレースホルダは arg1 で置き換えられ、次に小さい番号は arg2 で、以下同様です。 • **arg** - 任意の型で任意の数の引数
例	• `format('This %1 a %2','is', 'test')` → 'This is a test' • `format('This is %2','a bit unexpected but 2 is lowest number in string','normal')` → 'This is a bit unexpected but 2 is lowest number in string'

format_date

日付型または文字列をカスタム書式文字列でフォーマットします。Qt の date/time フォーマット文字列を使用します。詳細は QDateTime::toString を参照してください。

構文	format_date(datetime, format, [language]) 記号 [] は、オプションの引数を意味します。
引数	• **datetime** - 日付型、時間型または日付時刻型の値 • **format** - 文字列の書式設定に使用する文字列テンプレート

式	出力
d	十の位のゼロがない日付（1 から 31）
dd	十の位にゼロのある日付（01 から 31）
ddd	短縮形のローカル曜日名（例：' 月' から ' 日'）
dddd	長いローカル曜日名（例：' 月曜日' から ' 日曜日'）
M	十の位のゼロがない月（1 から 12）
MM	十の位にゼロのある月（01 から 12）
MMM	短縮形のローカル月名（例：'1 月' から '12 月'）
MMMM	長いローカル月名（例：'1 月' から '12 月'）
yy	2 桁の年（00 から 99）
yyyy	4 桁の年

以下の式は、書式文字列の時間部分に使用します。

式	出力
h	十の位のゼロなしの時（0 から 23、AM/PM 表示の場合は 1 から 12）
hh	十の位のゼロありの時（00 から 23、AM/PM 表示の場合は 01 から 12）
H	十の位のゼロなしの時（0 から 23、AM/PM 表示の場合も同様）
HH	十の位のゼロありの時（00 から 23、AM/PM 表示の場合も同様）
m	十の位のゼロなしの分（0 から 59）
mm	十の位のゼロありの分（00 から 59）
s	十の位のゼロなしの秒（0 から 59）
ss	十の位のゼロありの秒（00 から 59）
z	数字の後に付くゼロなしのミリ秒（0 から 999）
zzz	数字の後に付くゼロありのミリ秒（000 から 999）
AP or A	AM/PM 時間として解釈します。 *AP* は 'AM' または 'PM' のいずれかです。
ap or a	AM/PM 時間として解釈します。 *ap* は 'am' または 'pm' のいずれかです。

• **language** - 日付をカスタム文字列にフォーマットする際に使用する言語（2 文字または 3 文字の小文字、ISO 639 言語名コード ）。デフォルトでは現在の QGIS ユーザーのロケールを使用します

例

• `format_date('2012-05-15','dd.MM.yyyy')` → '15.05.2012'
• `format_date('2012-05-15','d MMMM yyyy','fr')` → '15 mai 2012'
• `format_date('2012-05-15','dddd')` → 'Tuesday'（現在のロケールが English 版の場合の結果）
• `format_date('2012-05-15 13:54:20','dd.MM.yy')` → '15.05.12'
• `format_date('13:54:20','hh:mm AP')` → '01:54 午後'

format_number

3 桁毎にロケールの桁区切り文字で区切られた数字を返します。デフォルトでは、現在の QGIS ユーザーのロケールが使用されます。また、指定された桁数で小数点以下の桁数をそろえます。

構文	format_number(number, [places=0], [language], [omit_group_separators=false], [trim_trailing_zeroes=false]) 記号 [] は、オプションの引数を意味します。
引数	• **number** - フォーマットする数値 • **places** - 小数点以下の桁数を表す整数 • **language** - 数値を文字列にフォーマットする際に使用する言語（2 文字または 3 文字の小文字、 ISO 639 言語名コード ）。デフォルトでは現在の QGIS ユーザーのロケールを使用します • **omit_group_separators** - true に設定すると、グループセパレータは文字列に含まれません • **trim_trailing_zeroes** - true を指定すると、小数点以下のゼロが文字列から取り除かれます
例	• `format_number(10000000.332,2)` → '10,000,000.33' （例えば現在のロケールが English 版の場合の結果） • `format_number(10000000.332,2,'fr')` → '10 000 000,33'

left

文字列の左から *n* 文字分の部分文字列を返します。

構文	left(string, length)
引数	• **string** - 文字列 • **length** - 整数値。文字列の左から数えた文字数
例	• `left('Hello World',5)` → 'Hello'

length

文字列の文字数、またはラインストリングジオメトリの長さを返します。

文字列バージョン

文字列の文字数を返します。

構文	length(string)
引数	• **string** - 文字数を数える文字列
例	• `length('hello')` → 5

ジオメトリバージョン

ラインジオメトリオブジェクトの長さを返します。計算は常にこのジオメトリの空間参照系（SRS）で平面的に計算され、長さの単位は SRS の単位と一致します。これは、プロジェクトの楕円体と距離単位設定にもとづいた楕円体計算が行われる $length 関数とは異なります。

構文	length(geometry)
引数	• **geometry** - ラインジオメトリオブジェクト
例	• `length(geom_from_wkt('LINESTRING(0 0, 4 0)'))` → 4.0

lower

文字列を小文字に変換します。

構文	lower(string)
引数	• **string** - 小文字に変換したい文字列
例	• `lower('HELLO World')` → 'hello world'

lpad

文字列の左側を指定文字で詰め、指定した幅にした文字列を返します。指定した幅が文字列の長さよりも小さい場合、文字列は切り詰められます。

構文	lpad(string, width, fill)
引数	• **string** - 幅を揃えたい文字列 • **width** - 新しい文字列の長さ • **fill** - 余白を詰める文字
例	• `lpad('Hello', 10, 'x')` → 'xxxxxHello' • `lpad('Hello', 3, 'x')` → 'Hel'

regexp_match

unicode 文字列内で正規表現にマッチする最初の位置を返します。部分文字列が見つからない場合、0 を返します。

構文	regexp_match(input_string, regex)
引数	• **input_string** - 正規表現に対してテストする文字列 • **regex** - テストする正規表現。バックスラッシュは二重にエスケープする必要があります（たとえば、空白文字に一致させる場合は "\\s" 、単語の境界に一致させる場合は "\\b" ）
例	• `regexp_match('QGIS ROCKS','\\sROCKS')` → 5 • `regexp_match('Budač','udač\\b')` → 2

regexp_replace

正規表現を用いて置き換えた文字列を返します。

構文	regexp_replace(input_string, regex, replacement)
引数	• **input_string** - 置換対象文字列 • **regex** - 置換する正規表現。バックスラッシュ文字は二重にエスケープする必要があります（例：空白文字に一致させるには "\\s" ） • **replacement** - 指定された正規表現のマッチ箇所を置き換える文字列。キャプチャされたグループは、 \\1 や \\2 などを使用して置換文字列内に挿入できます。
例	• `regexp_replace('QGIS SHOULD ROCK','\\sSHOULD\\s',' DOES ')` → 'QGIS DOES ROCK' • `regexp_replace('ABC123','\\d+','')` → 'ABC' • `regexp_replace('my name is John','(.*) is (.*)','\\2 is \\1')` → 'John is my name'

regexp_substr

正規表現にマッチする文字列の一部を返します。

構文	regexp_substr(input_string, regex)
引数	• **input_string** - 検索対象文字列 • **regex** - マッチさせる正規表現。バックスラッシュ文字は二重にエスケープする必要があります（例：空白文字に一致させるには "\\s" ）
例	• `regexp_substr('abc123','(\\d+)')` → '123'

replace

文字列を指定した別の文字列、配列またはマップ型オブジェクトで置換した文字列を返します。

文字列・配列バージョン

指定された文字列または文字列の配列を、別の文字列または文字列の配列で置換した文字列を返します。

構文	replace(string, before, after)
引数	• **string** - 入力文字列 • **before** - 置換対象の文字列または文字列の配列 • **after** - 代わりに使用する文字列または文字列の配列
例	• `replace('QGIS SHOULD ROCK','SHOULD','DOES')` → 'QGIS DOES ROCK' • `replace('QGIS ABC',array('A','B','C'),array('X','Y','Z'))` → 'QGIS XYZ' • `replace('QGIS',array('Q','S'),'')` → 'GI'

マップ型バージョン

与えられたマップ型を使用して、キーを対となる値で置き換えた文字列を返します。より長いキー文字列が最初に評価されます。

構文	replace(string, map)
引数	• **string** - 入力文字列 • **map** - キーと値のペアを含むマップ型オブジェクト
例	• `replace('APP SHOULD ROCK',map('APP','QGIS','SHOULD','DOES'))` → 'QGIS DOES ROCK' • `replace('forty two',map('for','4','two','2','forty two','42'))` → '42'

right

文字列の右から *n* 文字分の部分文字列を返します。

構文	right(string, length)
引数	• **string** - 文字列 • **length** - 整数値。文字列の右から数えた文字数
例	• `right('Hello World',5)` → 'World'

rpad

文字列の右側を指定文字で詰め、指定した幅にした文字列を返します。指定した幅が文字列の長さよりも小さい場合、文字列は切り詰められます。

構文	rpad(string, width, fill)
引数	• **string** - 幅を揃えたい文字列 • **width** - 新しい文字列の長さ • **fill** - 余白を詰める文字
例	• `rpad('Hello', 10, 'x')` → 'Helloxxxxx' • `rpad('Hello', 3, 'x')` → 'Hel'

strpos

文字列中で部分文字列が最初にマッチする位置を返します。部分文字列が見つからなければ、0 を返します。

構文	strpos(haystack, needle)
引数	• **haystack** - 対象文字列 • **needle** - 検索する文字列
例	• `strpos('HELLO WORLD','WORLD')` → 7 • `strpos('HELLO WORLD','GOODBYE')` → 0

substr

文字列の一部分を返します。

構文	substr(string, start, [length]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - 入力文字列 • **start** - 抽出開始位置を表す整数。1 から数える。負の値を指定した場合、文字列の最後から数えた位置から開始します • **length** - 抽出する文字列の長さを表す整数。負の値を指定した場合、文字列の末尾から指定された長さ分だけ文字が除かれた文字列を返します。
例	• `substr('HELLO WORLD',3,5)` → 'LLO W' • `substr('HELLO WORLD',6)` → ' WORLD' • `substr('HELLO WORLD',-5)` → 'WORLD' • `substr('HELLO',3,-1)` → 'LL' • `substr('HELLO WORLD',-5,2)` → 'WO' • `substr('HELLO WORLD',-5,-1)` → 'WORL'

title

文字列の全ての単語をタイトルケース（先頭が大文字で後に小文字が続く）に変換します。

構文	title(string)
引数	• **string** - タイトルケースに変換したい文字列
例	• `title('hello WOrld')` → 'Hello World'

to_string

数値を文字列に変換します。

構文	to_string(number)
引数	• **number** - 整数または実数値。文字列に変換する数値
例	• `to_string(123)` → '123'

trim

文字列の先頭と末尾から全てのホワイトスペース（空白文字、タブ等）を削除します。

構文	trim(string)
引数	• **string** - 空白を削除したい文字列
例	• `trim(' hello world ')` → 'hello world'

upper

文字列を大文字に変換します。

構文	upper(string)
引数	• **string** - 大文字に変換したい文字列
例	• `upper('hello WOrld')` → 'HELLO WORLD'

wordwrap

指定した最大・最小文字数に合わせて改行した文字列を返します。

構文	wordwrap(string, wrap_length, [delimiter_string]) 記号 [] は、オプションの引数を意味します。
引数	• **string** - ワードラップしたい文字列 • **wrap_length** - 整数。 wrap_length が正の数値の場合、理想的な一行あたりの最大文字数を意味します。負の値の場合、この数字は改行する最小文字数を意味します。 • **delimiter_string** - （オプション）改行を行う区切り文字
例	• `wordwrap('UNIVERSITY OF QGIS',13)` → 'UNIVERSITY OF QGIS' • `wordwrap('UNIVERSITY OF QGIS',-3)` → 'UNIVERSITY OF QGIS'

13.2.25 ユーザー式

このグループには、 **ユーザー保存式** として保存された式が含まれます。

13.2.26 変数

このグループには、アプリケーションやプロジェクトファイル、その他の設定に関連した動的な変数が含まれます。変数が使用できるかどうかは、利用される状況によって異なります。

-  式による地物選択 ダイアログから使用
-  フィールド計算機 ダイアログから使用
- レイヤプロパティダイアログから使用
- 印刷レイアウトから使用

式の中でこれらの変数を使用するには、変数の先頭に `@` 文字を付ける必要があります（例： `@row_number` ）

変数	説明
algorithm_id	アルゴリズムのユニーク ID
animation_end_time	アニメーションの時間範囲全体の終了日時（datetime 値）
animation_interval	アニメーションの時間範囲全体の長さ（interval 値）
animation_start_time	アニメーションの時間範囲全体の開始日時（datetime 値）
atlas_feature	現在の地図帳地物（地物オブジェクト）
atlas_featureid	現在の地図帳地物 ID
atlas_featurenumber	レイアウト内の現在の地図帳地物の番号
atlas_filename	現在の地図帳のファイル名
atlas_geometry	現在の地図帳地物のジオメトリ
atlas_layerid	現在の地図帳カバレッジレイヤの ID
atlas_layername	現在の地図帳カバレッジレイヤのレイヤ名
atlas_pagename	現在の地図帳のページ名
atlas_totalfeatures	地図帳地物の総数
canvas_cursor_point	キャンバス上の最後のカーソル位置（プロジェクトの地理座標）
cluster_color	クラスタ内のシンボルの色。シンボルに色が混在している場合は NULL
cluster_size	クラスタ内に含まれるシンボル数
current_feature	属性フォームまたはテーブル行で現在編集中の地物
current_geometry	属性フォームまたはテーブル行で現在編集中の地物のジオメトリ
current_parent_feature	親フォームで現在編集中の地物を表す変数。埋め込みフォームでのみ使用可能。
current_parent_geometry	親フォームで現在編集中の地物のジオメトリを表す変数。埋め込みフォームでのみ使用可能。
form_mode	フォームを使う対象。文字列で AddFeatureMode, SingleEditMode, MultiEditMode, SearchMode, AggregateSearchMode または IdentifyMode のいずれか。
feature `NEW in 3.28`	評価されている現在の地物。この関数を 'attribute' 関数と共に使用すると、現在の地物の属性値を評価できます。

次のページに続く

表 13.1 – 前のページからの続き

変数	説明
frame_duration	各アニメーションフレームの時間の長さ（interval 値）
frame_number	アニメーション再生時の現在のフレーム番号
frame_rate	アニメーション再生時の 1 秒あたりフレーム数
fullextent_maxx	全キャンバス領域での x 最大値（全レイヤ）
fullextent_maxy	全キャンバス領域での y 最大値（全レイヤ）
fullextent_minx	全キャンバス領域での x 最小値（全レイヤ）
fullextent_miny	全キャンバス領域での y 最小値（全レイヤ）
geometry `NEW in 3.28`	評価されている地物のジオメトリ
geometry_part_count	レンダリングされる地物のジオメトリパートの数
geometry_part_num	レンダリングされる地物の現在のジオメトリパートの番号
geometry_point_count	レンダリングされるジオメトリ部分のポイントの数
geometry_point_num	レンダリングされるジオメトリ部分の現在のポイントの番号
geometry_ring_num	レンダリングされる地物の現在のジオメトリのリング番号（ポリゴン地物のみ）。外側リングの番号は 0 です。
grid_axis	現在のグリッド注記の軸（経度は'x' 、緯度は'y' ）
grid_number	現在のグリッド注記の値
id `NEW in 3.28`	評価されている現在の地物の ID
item_id	レイアウトアイテムのユーザ ID（ユニークとは限らない）
item_uuid	レイアウトアイテムのユニーク ID
layer	現在のレイヤ
layer_crs	現在のレイヤの CRS の Authority ID
layer_id	現在のレイヤ ID
layer_ids	現在のプロジェクト内の全てのレイヤ ID のリスト
layer_name	現在のレイヤ名
layers	現在のプロジェクト内の全てのレイヤのリスト
layout_dpi	印刷レイアウトの解像度（DPI）
layout_name	印刷レイアウトの名前
layout_numpages	印刷レイアウトのページ数
layout_page	印刷レイアウト内の現在のアイテムのページ番号
layout_pageheight	レイアウトのアクティブなページの高さ（標準のページサイズの場合は mm 単位、カスタムのページサイズの場合は使用されている単位）
layout_pageoffsets	各ページの上端の Y 座標を要素とする配列。ページのサイズが変化することがある場合に、この変数を使用することでページ上にアイテムを動的に配置できます。
layout_pagewidth	レイアウトのアクティブなページの幅（標準のページサイズの場合は mm 単位、カスタムのページサイズの場合は使用されている単位）
legend_column_count	凡例のカラム数
legend_filter_by_map	凡例の内容が地図の範囲と連動するかどうか
legend_filter_out_atlas	凡例の内容が地図帳の範囲と連動するかどうか
legend_split_layers	凡例のレイヤアイテムが複数列に分割される可能性があるかどうか
legend_title	凡例のタイトル
legend_wrap_string	凡例のテキストを改行する文字（文字列）
map_crs	現在の地図の座標参照系

次のページに続く

表 13.1 – 前のページからの続き

変数	説明
map_crs_acronym	現在の地図の座標参照系の頭字語
map_crs_definition	現在の地図の座標参照系の完全な定義
map_crs_description	現在の地図の座標参照系の名前
map_crs_ellipsoid	現在の地図の座標参照系の回転楕円体
map_crs_proj4	現在の地図の座標参照系の proj4 定義
map_crs_projection	地図の CRS で使用される投影法の記述名（例： 'Albers Equal Area' ）
map_crs_wkt	現在の地図の座標参照系の WKT 定義
map_end_time	地図の時系列範囲の終了値（datetime 値）
map_extent	現在の地図の範囲を表すジオメトリ
map_extent_center	地図範囲の中心を表すポイント地物
map_extent_height	地図の現在の高さ
map_extent_width	地図の現在の幅
map_id	現在の地図の出力対象の ID。スクリーンの場合は canvas、印刷レイアウトの場合はアイテム ID
map_interval	地図の時系列範囲の長さ（interval 値）
map_layer_ids	地図で表示中のレイヤ ID のリスト
map_layers	地図で表示中のレイヤのリスト
map_rotation	地図の現在の回転
map_scale	地図の現在の縮尺
map_start_time	地図の時系列範囲の開始値（datetime 値）
map_units	地図の距離単位
model_path	現在のモデルの（ファイル名を含んだ）フルパス（または、モデルがプロジェクトに埋め込まれている場合にはプロジェクトのパス）
model_folder	現在のモデルがあるフォルダ（または、モデルがプロジェクトに埋め込まれている場合にはプロジェクトのフォルダ）
model_name	現在のモデルの名前
model_group	現在のモデルのグループ
notification_message	データプロバイダから送信された通知メッセージの内容（プロバイダからの通知によってトリガされるアクションでのみ利用可）
parent	**集約** 関数でフィルタリングする際に、親レイヤの現在の地物を参照しその属性とジオメトリにアクセスできる変数
project_abstract	プロジェクトのメタデータより取得した、プロジェクトの要約
project_area_units	ジオメトリの面積計算で使用する、現在のプロジェクトの面積単位
project_author	プロジェクトのメタデータより取得した、プロジェクトの制作者
project_basename	現在のプロジェクトのファイル名のベース名（パスと拡張子なしの名前）
project_creation_date	プロジェクトのメタデータより取得した、プロジェクトの作成日
project_crs	プロジェクトの座標参照系
project_crs_arconym	プロジェクトの座標参照系の頭字語
project_crs_definition	プロジェクトの座標参照系の完全な定義
project_crs_description	プロジェクトの座標参照系の名前
project_crs_ellipsoid	プロジェクトの座標参照系の回転楕円体
project_crs_proj4	プロジェクトの座標参照系の proj4 定義
project_crs_wkt	プロジェクトの座標参照系の WKT（well known text）定義

次のページに続く

表 13.1 – 前のページからの続き

変数	説明
project_distance_units	ジオメトリの長さや距離計算で使用する、現在のプロジェクトの距離単位
project_ellipsoid	ジオメトリの測地面積や測地距離の計算で使用する、現在のプロジェクトの回転楕円体の名前
project_filename	現在のプロジェクトのファイル名
project_folder	現在のプロジェクトのフォルダ
project_home	現在のプロジェクトのホームパス
project_identifier	プロジェクトのメタデータより取得した、プロジェクトの識別子
project_keywords	プロジェクトのメタデータより取得した、プロジェクトのキーワード
project_last_saved	プロジェクトが最後に保存された日時
project_path	現在のプロジェクトのフルパス（ファイル名を含む）
project_title	現在のプロジェクトのタイトル
project_units	プロジェクト CRS の長さ単位
qgis_locale	現在の QGIS の言語ロケール
qgis_os_name	現在の OS 名（例： 'windows' 、 'linux' 、 'osx' ）
qgis_platform	QGIS プラットフォーム： 'desktop' または 'server'
qgis_release_name	現在の QGIS リリース名
qgis_short_version	現在の QGIS バージョンの短縮文字列
qgis_version	現在の QGIS バージョンの文字列
qgis_version_no	現在の QGIS バージョンの番号
row_number	現在の行の番号を格納します
snapping_results	地物のデジタイジング中にスナップした結果へのアクセスを提供します（地物の追加時のみ利用可）
scale_value	現在のスケールバーの距離の値
selected_file_path	外部ストレージシステムにファイルをアップロードする際に、ファイルセレクタウィジェットによって選択されたファイルのパス
symbol_angle	地物の描画に使用されるシンボルの角度（マーカーシンボルのみで有効）
symbol_color	地物の描画に使用されるシンボルの色
symbol_count	このシンボルで表現されている地物の数（印刷レイアウトの凡例でのみ有効）
symbol_id	シンボルの内部 ID（印刷レイアウトの凡例でのみ有効）
symbol_label	シンボルのラベル（ユーザー定義ラベルもしくはデフォルトの自動生成ラベル。印刷レイアウトの凡例でのみ有効）
symbol_layer_count	シンボル内のシンボルレイヤの総数
symbol_layer_index	現在のシンボルレイヤのインデックス
symbol_marker_column	マーカーの列数（ポイントパターン塗りつぶしでのみ有効）
symbol_marker_row	マーカーの行数（ポイントパターン塗りつぶしでのみ有効）
user_account_name	現在のユーザーの OS アカウント名
user_full_name	現在のユーザーの OS ユーザー名
value	現在の値
vector_tile_zoom	レンダリングされている地図のベクトルタイルの正確なズームレベル（現在の地図の縮尺から導出される）。通常は [0, 20] の間の値。@zoom_level とは異なり、この変数は浮動小数点値であり、2 つの整数のズームレベルの間の値を補間するために使用できます。

次のページに続く

表 13.1 – 前のページからの続き

変数	説明
with_variable	式の中で使用するための変数を設定することで、同じ値を繰り返し計算することを避けられます。
zoom_level	レンダリングされている地図のベクトルタイルのズームレベル（現在の地図の縮尺から導出される）。通常は [0, 20] の間の値。

例

- 印刷レイアウトにおいて、地図アイテムの中心の X 座標を返します:

```
x( map_get( item_variables( 'map1'), 'map_extent_center' ) )
```

- 現在のレイヤの各地物について、自身と交差する airport 地物の数を返します:

```
aggregate( layer:='airport', aggregate:='count', expression:="code",
               filter:=intersects( $geometry, geometry( @parent ) ) )
```

- ラインの最初にスナップしたポイントの object_id を取得します:

```
with_variable(
  'first_snapped_point',
  array_first( @snapping_results ),
  attribute(
    get_feature_by_id(
      map_get( @first_snapped_point, 'layer' ),
      map_get( @first_snapped_point, 'feature_id' )
    ),
    'object_id'
  )
)
```

13.2.27 最近の関数

このグループには、最近使った関数が含まれます。関数の利用のコンテクスト（地物選択、フィールド計算、一般など）に応じて、最近適用した式は対応するリスト（上限 10 個）に追加され、最近使ったものから順にリストに並びます。これにより、以前に使用した関数を素早く探し出して再適用することが簡単にできます。

第14章　スタイルライブラリ

14.1 スタイルマネージャ

14.1.1 スタイルマネージャダイアログ

スタイルマネージャ は汎用スタイルアイテムを管理し、作成する場所です。汎用スタイルアイテムにはシンボル、カラーランプ、テキスト形式、ラベルの設定があり、地物やレイヤ、印刷レイアウトのシンボル設定に使用することができます。これらはアクティブな **ユーザープロファイル** の下にある `symbology-style.db` データベースに保存され、そのプロファイルで開かれた全てのプロジェクトファイル共有されます。スタイルアイテムは **スタイルマネージャ** ダイアログのインポートとエクスポート機能を使って他の人と共有することもできます。

このモードレスダイアログは、以下の方法で開くことができます。

- **設定** **スタイルマネージャ...** メニュー
- プロジェクトツールバーの スタイルマネージャ ボタン
- ベクタレイヤの **レイヤのプロパティ** メニュー（の **シンボルの設定** あるいは **テキストの書式設定**）にある スタイルマネージャ ボタン

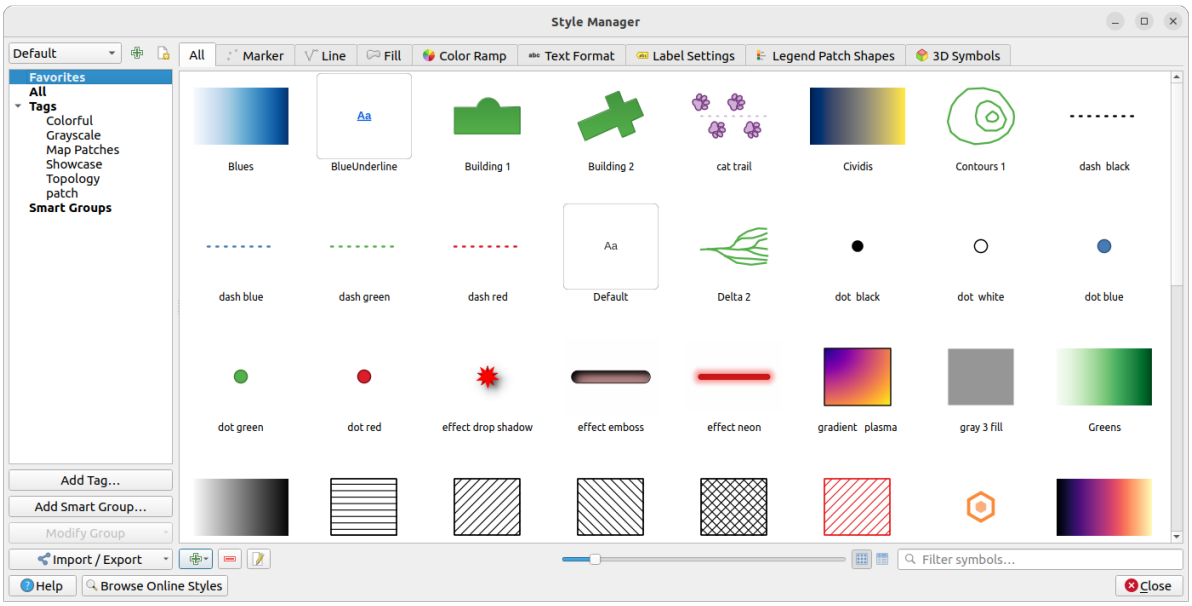


図 14.1: スタイルマネージャ

スタイルアイテムの整理

スタイルマネージャ ダイアログでは、プレビューアイテムがタブに整理されたフレームが中央に表示されます。

- **すべて** は、マーカー、ライン、塗りつぶしシンボルやラベルの設定、定義済みカラーランプやテキスト形式の完全なコレクションです。
- **マーカー** は、ポイントシンボルのみを表示します
- **ライン** はラインシンボルのみを表示します
- **塗りつぶし** は塗りつぶしシンボルのみを表示します
- **カラーランプ**
- **テキスト形式** は、テキストのフォント、色、バッファ、影、背景（つまりは、例えばレイアウトで使用されるラベルの書式設定部分のすべて）を格納する **テキストのフォーマット** を管理します。
- **ラベルの設定** は、テキストの書式や、ラベル配置、優先度、引出し線、レンダリングなどレイヤの種類に固有の設定を含む **ラベルの設定** を管理します。
- **凡例パッチ** は、カスタムの凡例パッチを管理します。これには、ジオメトリの **マーカー** 、 **ライン** そして **塗りつぶし** が含まれます。
- *3D* **シンボル** は、 *3D* **マップビュー** において地物をレンダリングするために、 *3D* **ビュープロパティ** （押出、シェーディング、高さ、...）を持つシンボルを設定します。

スタイルは、右下の端にある **アイコンビュー** や **リストビュー** ボタンで並べることができます。どちらのビューでも、ツールチップにはそのスタイルのインスタンスが大きく表示されます。アイコンの左にあるサムネイルサイズのスライダーを使用すると、ダイアログ内の実際のサムネイルのサイズを調節し、シンボルのプレビューを見やすくできます。

各アイテム群については、左側のパネルに表示されている様々なカテゴリに要素を整理することができます：

- **お気に入り** ：アイテムを設定する際ににデフォルトで表示され、拡張可能なアイテムのセットを表示します。
- **すべて** ：アクティブなタイプで利用可能な全てのアイテムをリスト表示します。
- **タグ** ：アイテムを識別するために使用することができるラベルのリストを表示します。アイテムには複数のタグを付けることが可能です。リストのタグを選択すると、タブはそのタグに属しているアイテムだけが表示されるように更新されます。新しいタグを作成して後でアイテムのセットに付けられるようにするには、 **タグを追加**... ボタンを押すか、任意のタグのコンテキストメニューから **タグを追加**... を選択します。
- **スマートグループ** ：スマートグループは、設定された条件に応じてシンボルを動的に取得します（例：図 14.2 ）。スマートグループを作成するには、 **スマートグループを追加**... ボタンをクリックします。ダイアログボックスでは、選択するアイテムをフィルタリング（特定のタグを持っている、名前の一部が文字列に一致する、など）するための式を入力することができます。入力された条件を満足するシンボルやカラーランプ、テキスト形式、ラベルの設定は、自動的にスマートグループに追加されます。

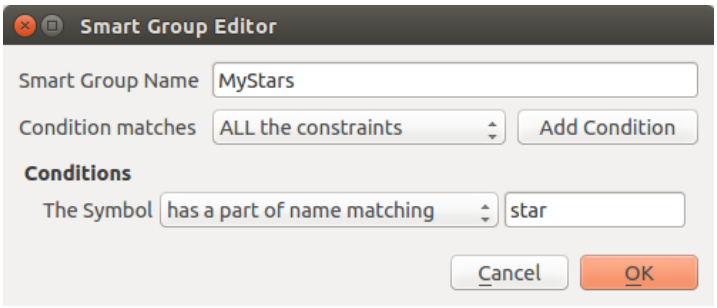


図 14.2: スマートグループを作成する

タグとスマートグループは相互排他的ではありません。これらはスタイル要素を整理するための、単なる 2 つの異なる方法です。入力した制約条件に基づいて属するアイテムが自動的に取得されるスマートグループとは異なり、タグはユーザーによって付けられます。これらのカテゴリを編集するには、次のいずれかの方法で行います：

- アイテムを選択し、右クリックして **タグに追加** を選び、タグ名を選択するか、新しいタグを作成します。
- タグを選択し、 **グループの変更...** **シンボルに選択したタグ付ける** を押します。各アイテムの隣にチェックボックスが現れ、アイテムを選択するか除外するかを設定できます。選択が完了したら、**グループの変更...** **タグ付けの完了** を押します。
- スマートグループを選択し、 **グループの変更...** **スマートグループの編集...** を押すと、 **スマートグループエディタ** ダイアログで新しい制約条件を設定することができます。このオプションは、スマートグループのコンテキストメニューからも利用可能です。

タグやスマートグループを削除するには、これを右クリックして **削除** ボタンを選択します。これは、カテゴリにグループ化されたアイテムを削除するのではないことに留意してください。

アイテムの追加、編集、削除

上で述べたように、スタイル要素は様々なタブの下にリストされます。リストの内容はアクティブなカテゴリ（タグ、スマートグループ、お気に入り...）に依存します。 タブが有効になっているときは、以下の操作ができます。

- 新しいアイテムの追加：アイテム追加 ボタンを押し、 **シンボル** 、 **カラーランプ** または **テキスト形式やラベルの設定** ビルダーの説明に従ってアイテムを設定します。
- 既存のアイテムの編集：アイテムを選択し、アイテム編集 ボタンを押して、上で述べたのと同様にアイテムの設定を行います。
- 既存のアイテムを削除する：不要になった要素を削除するには、それを選択して アイテム削除 （右クリックからも利用可能）ボタンを押します。そのアイテムはローカルデータベースから削除されます。

全てのプラグイン タブでは、全ての種類のアイテムについてこれらのオプションにアクセスできることに留意してください。

アイテムを選択して右クリックすると、以下の操作もできます。

- **お気に入りに追加**
- **お気に入りから削除**
- **タグに追加**　を選び、適切なタグを選択するか、使用する新しいタグを作成します。現在指定されているタグはチェック済みになっています
- **タグをクリアする** ：シンボルから全てのタグを外します
- **アイテムを削除**
- **アイテムを編集** ：右クリックしたアイテムに適用されます
- **アイテムのコピー**
- **アイテムを貼り付け**... ：スタイルマネージャのカテゴリの一つもしくは QGIS の他の場所（シンボルや色ボタン等）に貼りつけます
- **選択したシンボルを** *PNG* **としてエクスポート**...（シンボルのみ利用可能）
- **選択したシンボルを** *SVG* **としてエクスポート**...（シンボルのみ利用可能）

スタイルアイテムの共有

スタイルマネージャダイアログの左下にある **インポートとエクスポート** ツールには、シンボル、カラーランプ、テキスト形式、ラベルの設定を他者と簡単に共有するためのオプションがあります。これらのオプションは、アイテム上で右クリックして使用することもできます。

アイテムのエクスポート

アイテムのセットを `.XML` ファイルにエクスポートすることができます。

1. **インポートとエクスポート** ドロップダウンメニューを展開し、 **アイテムのエクスポート**... を選択します。
2. エクスポートにまとめたいアイテムを選択します。選択はマウスを使用するか、事前に設定したタグやグループを使用して行うことができます。
3. 準備ができたら **エクスポート** ボタンを押します。ファイルの保存先を指定するように求められます。XML 形式は、選択したアイテム全てを含む単一のファイルを生成します。このファイルは、他のユーザーのスタイルライブラリにインポートできます。

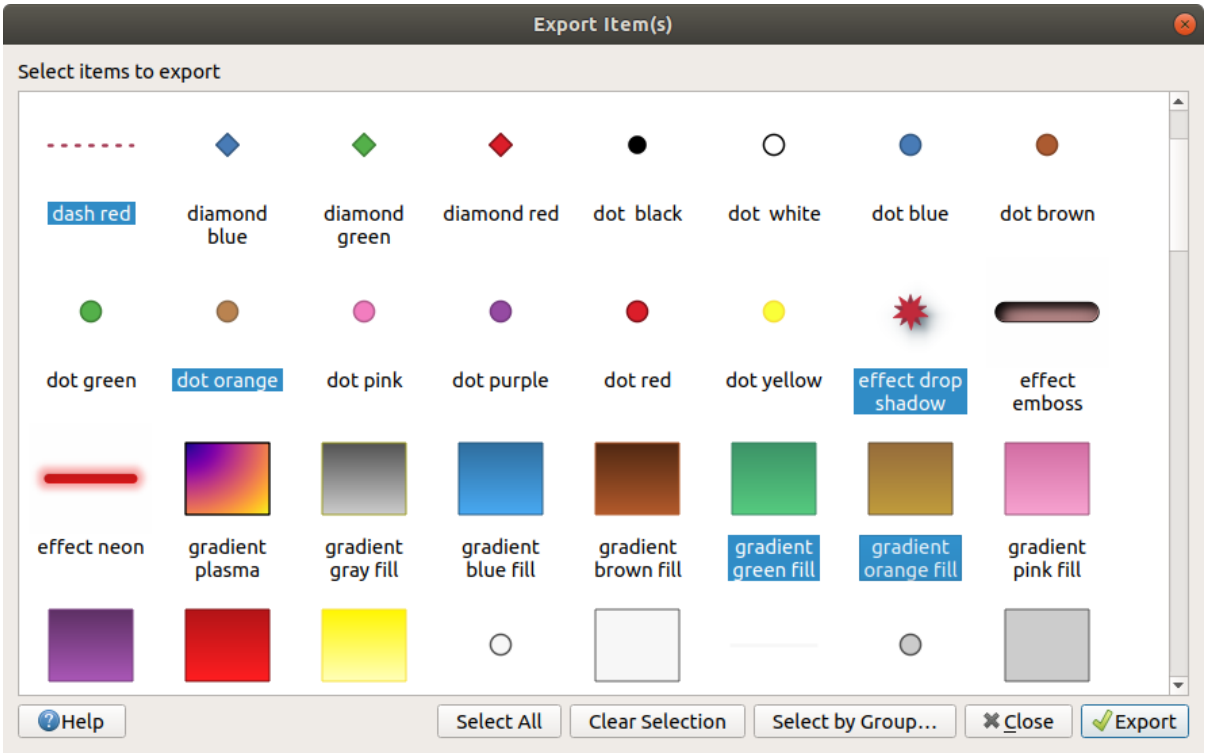


図 14.3: スタイルアイテムのエクスポート

シンボルを選択した場合には、 .PNG または .SVG 形式でエクスポートすることもできます。 .PNG または .SVG （どちらも他のスタイルアイテムタイプでは利用不可です）へのエクスポートは、指定されたフォルダ内に選択された各シンボルに対するファイルを作成します。SVG フォルダを他のユーザーの **設定　オプション　システム** メニューの *SVG* **パス** に追加することで、これらすべてのシンボルに直接アクセスできるようになります。

アイテムのインポート

新しいアイテムをインポートすることで、スタイルライブラリを拡張できます。

1. ダイアログの左下にある **インポートとエクスポート** ドロップダウンメニューを展開し、 **アイテムのインポート** ボタンを押します。
2. 新しく開いたダイアログで、スタイルアイテムのソース（ディスク上の .xml ファイルまたは URL）を指定します。
3. インポートするアイテムを **お気に入りに追加** するかを設定します。
4. **埋め込みタグをインポートしない** をチェックすると、インポートするアイテムに関連付けられたタグをインポートしません。
5. 新しいアイテムに適用する **追加のタグ** 名を指定します。
6. ライブラリに追加するシンボルをプレビューから選択します。
7. 最後に、 **インポート** ボタンを押します。

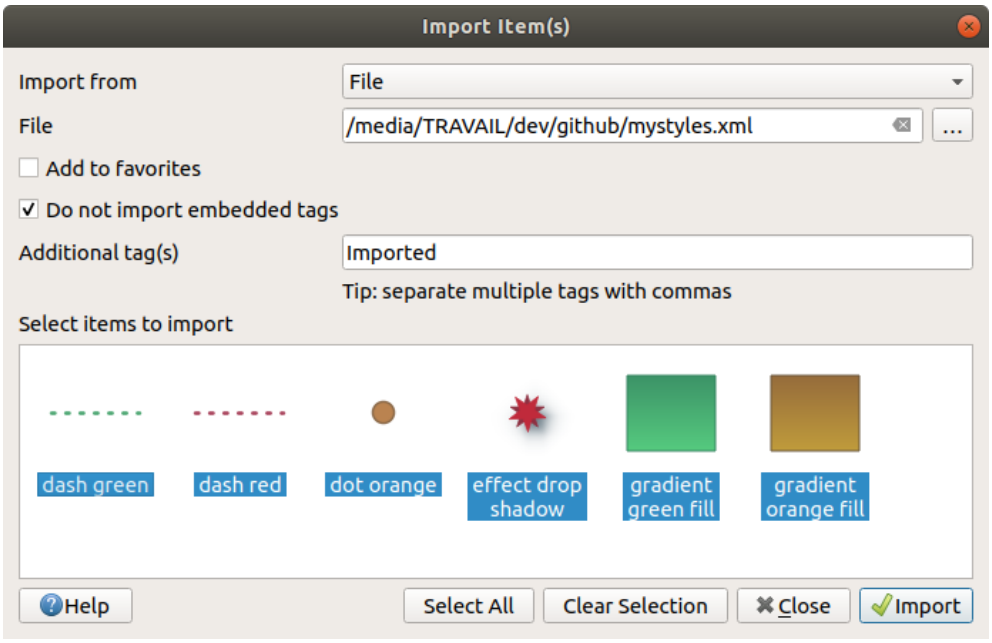


図 14.4: スタイルアイテムのインポート

ブラウザパネルを使用する

ブラウザ パネルから直接、アクティブなユーザープロファイルのスタイルデータベースにスタイルアイテムをインポートすることも可能です。

1. スタイルの `.xml` ファイルをブラウザで選択します。
2. マップキャンバス上へドラッグアンドドロップするか、右クリックして **スタイルのインポート**... を選びます。
3. **アイテムのインポート** に従い、 **アイテムのインポート** ダイアログを入力します。
4. **インポート** ボタンを押すと、選択されたスタイルアイテムがスタイルデータベースに追加されます。

ブラウザ内でスタイルファイルをダブルクリックすると、 **スタイルマネージャ** ダイアログが開き、ファイル内にあるアイテムが表示されます。アイテムを選択して **デフォルトスタイルにコピー**... を押すと、それらをアクティブなスタイルデータベースにインポートすることができます。アイテムにタグを付けることも可能です。また、このダイアログはスタイルファイルを右クリックして **スタイルを開く**... コマンドからも利用可能です。

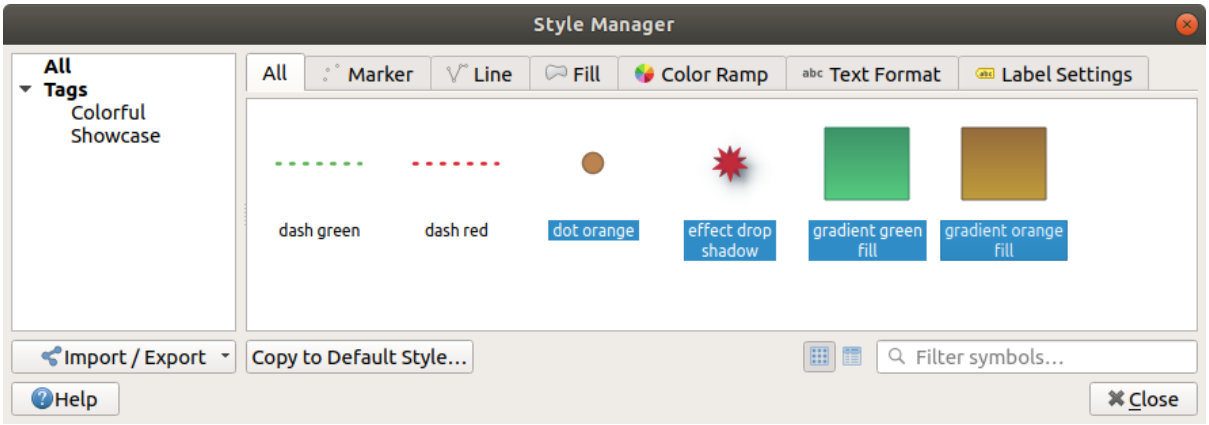


図 14.5: スタイルアイテムファイルを開く

このダイアログでは、単一のシンボルを .PNG または .SVG 形式のファイルとしてエクスポートすることもできます。

オンラインリポジトリを使用する

QGIS プロジェクトでは、QGIS ユーザーによって共有されたスタイルのコレクションリポジトリを管理しています。これは https://plugins.qgis.org/styles にあり、 **スタイルマネージャ** ダイアログの下部にある **オンラインスタイルを表示** ボタンを押すことでアクセスできます。

このリポジトリのスタイルを利用するには、

1. 任意のスタイルアイテムをブラウズしたり、種類や名前に基づいて検索します
2. スタイルファイルをダウンロードし、展開します
3. 上で述べた任意のインポート方法で、 .xml ベースのファイルを QGIS のスタイルデータベースに読み込みます。

14.1.2 カラーランプの設定

スタイルマネージャ ダイアログの「カラーランプ」タブでは、左のパネル内で選択されたカテゴリに基づくさまざまなカラーランプをプレビューすることができます。

カスタムカラーランプを作成するには、カラーランプタブをアクティブにして アイテム追加 ボタンをクリックします。ボタンをクリックすると、カラーランプのタイプを選ぶドロップダウンリストが表示されます：

- **勾配グリッド**（*Gradient*）：開始色と終了色を指定し、 **連続的** または **離散値** カラーランプを生成します。ランププレビューをダブルクリックすると、好きなだけ中間カラーストップを追加できます。カラーストップインジケータ上をクリックして、下部にある **グラデーションストップ** で以下の設定ができます:
  - カラーランプの始点からの **相対位置** の調整。インジケータをマウスでドラッグしたり、矢印キーを押す（ Shift キーを押しながらだと大きく移動）ことでも調整ができます。

- 色どうしの補間に使用する色モデルの指定： *RGB* 、 *HSL* 、 *HSV* のいずれかです。状況によっては、このオプションを使うと中間調の彩度が落ちるのを防げ、より美しいグラデーションが得られます。
- *HSL* または *HSV* の色モデル指定の場合の **Hue** 要素に対して、補間方向を設定します。値は **時計回り** または **反時計まわり** です。
- **色プロパティ** の設定
- **ストップを削除** ボタンまたは DEL キーを押して、カラーストップを削除

出力のプロット グループは、カラーストップの位置や不透明度、HSL 成分を変更することで、カラーランプをデザインするためのもう一つのグラフィカルな方法を提供します。

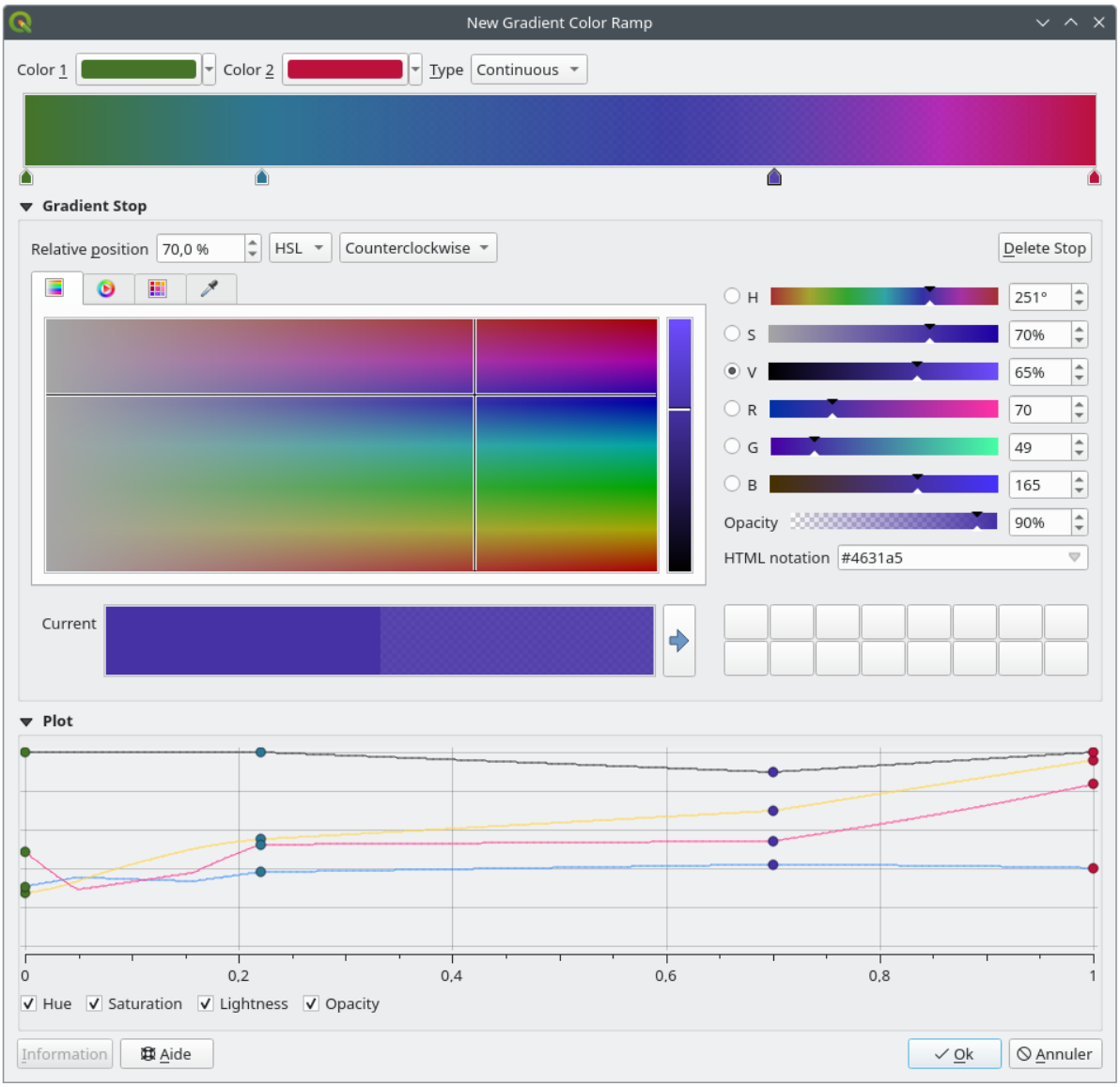


図 14.6: 複数のストップを持つカスタムグラデーションカラーランプの例

ヒント: カラースポットからグラデーションランププレビューに色をドラッグ＆ドロップすると、新

しいカラーストップを追加します。

- **カラープリセット** ：ユーザーが選択した色のリストで構成されるカラーランプを作成します。
- **ランダム** ： **色相** 、 **彩度** 、 **値** の範囲と、 **不透明度** と色の数（ **クラス** ）に基づいてランダムな色のセットを作成します。
- **カタログ**: *ColorBrewer* ：事前に定義された離散的なカラーグラデーションのセットで、カラーランプの色数をカスタマイズできます。
- **カタログ**: *cpt-city* ： **標準グラデーション** としてローカルに保存するためのカラーグラデーションのカタログ全体へのアクセスです。cpt-city オプションは、追加設定の必要のないものも含め、何百ものテーマがある新しいダイアログを開きます。

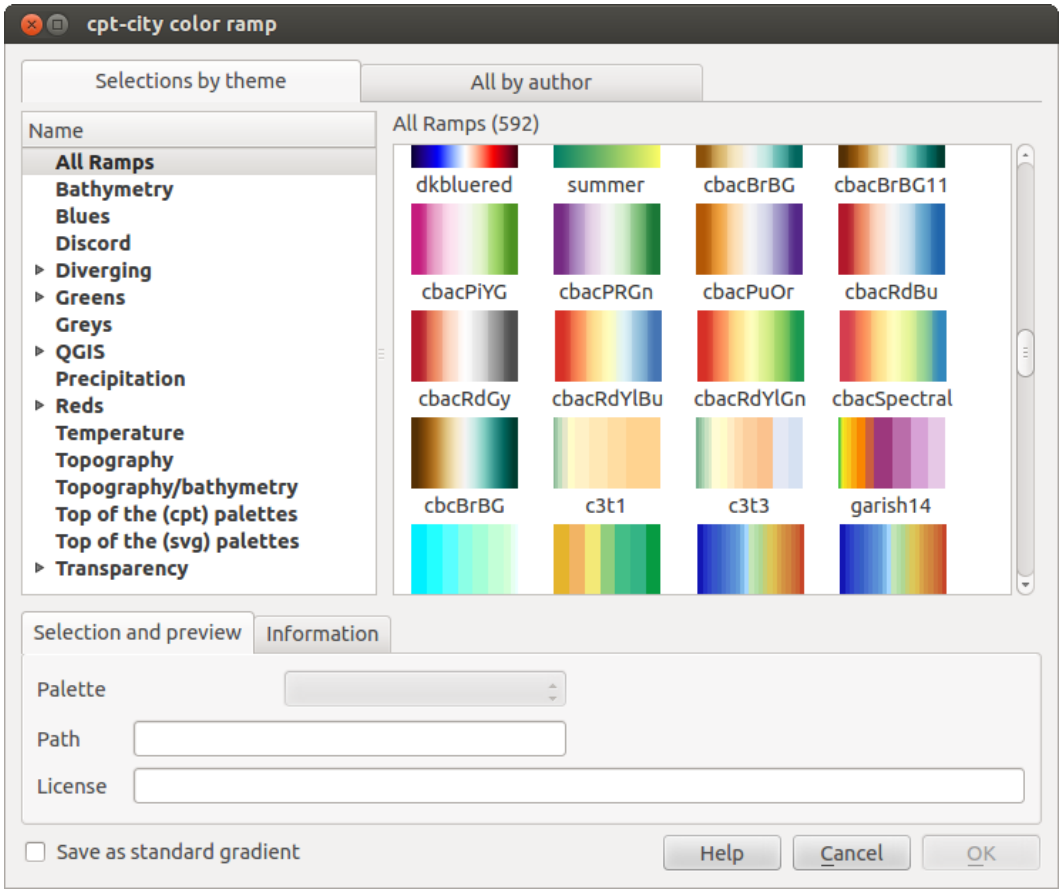


図 14.7: 何百ものカラーランプがある cpt-city ダイアログ

14.1.3 凡例パッチの作成

凡例パッチを作成するには、 **凡例パッチ** タブを有効にして アイテム追加 ボタンをクリックします。ボタンをクリックすると、ジオメトリの種類を選択するためのドロップダウンリストが現れます：

- **マーカー凡例パッチを保存**... ：ポイントジオメトリに使用します
- **ライン凡例パッチを保存**... ：ラインジオメトリに使用します
- **塗りつぶし凡例パッチを保存**... ：ポリゴンジオメトリに使用します

3 つのオプションのいずれも、同じダイアログが表示されます。

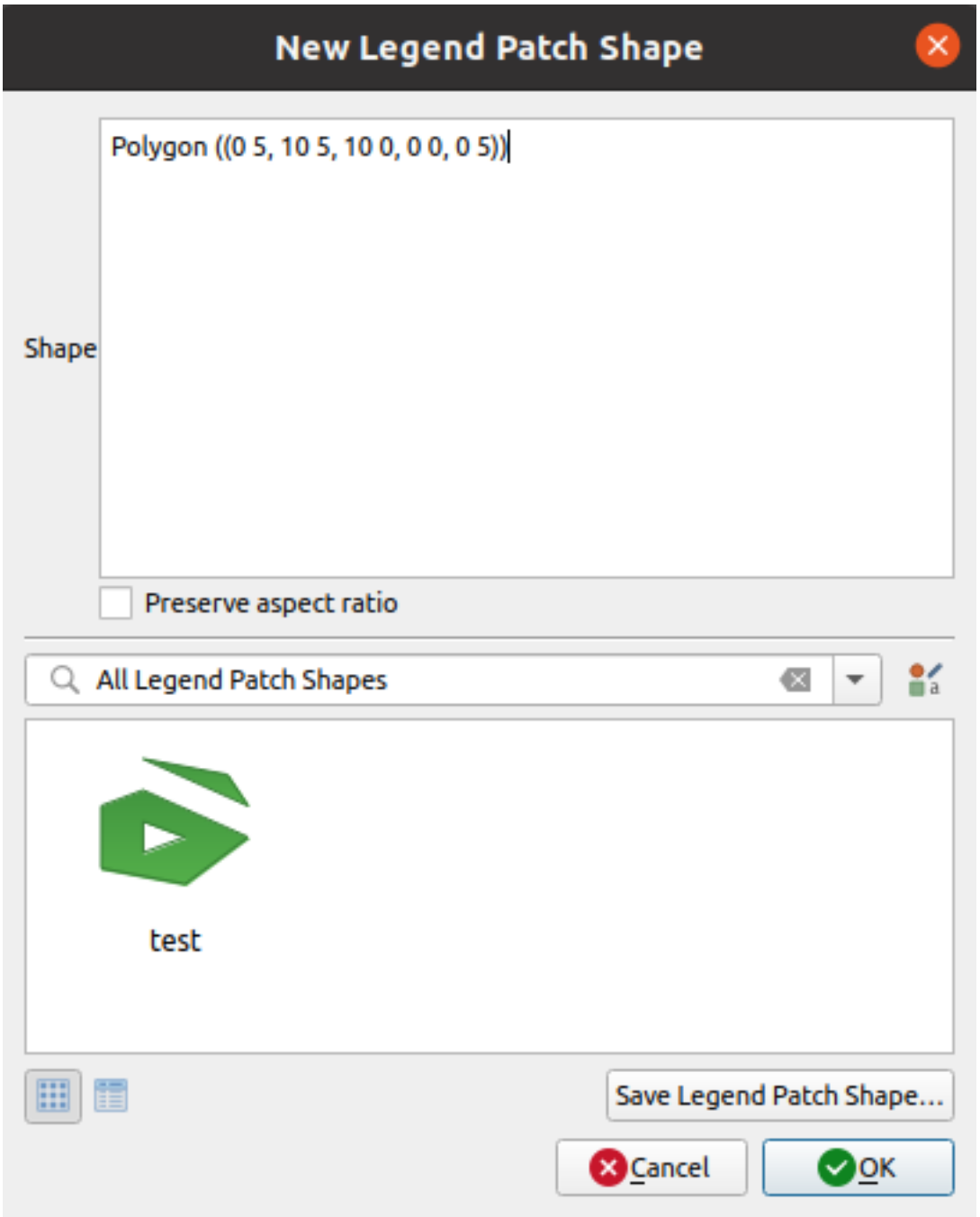


図 14.8: 新規凡例パッチの作成

ダイアログは同じですが、選択したジオメトリの種類によって shape の種類と表示される凡例パッチの shape のみが異なります。以下のオプションが利用可能です。

- *Shape* ：凡例パッチ図形の形状を WKT 文字列で定義します。シングルパートとマルチパートのジオメトリを使用できますが、GeometryCollection は使用できません。
- **アスペクト比を維持**
- 利用可能な凡例パッチの **アイコンビュー** または **リストビュー** 。タグでフィルタリングできます。

新しい Shape を定義したら、 **凡例パッチの形を保存**... を押すか、 *OK* ボタンを押します。どちらでも、同じダイアログが開きます。

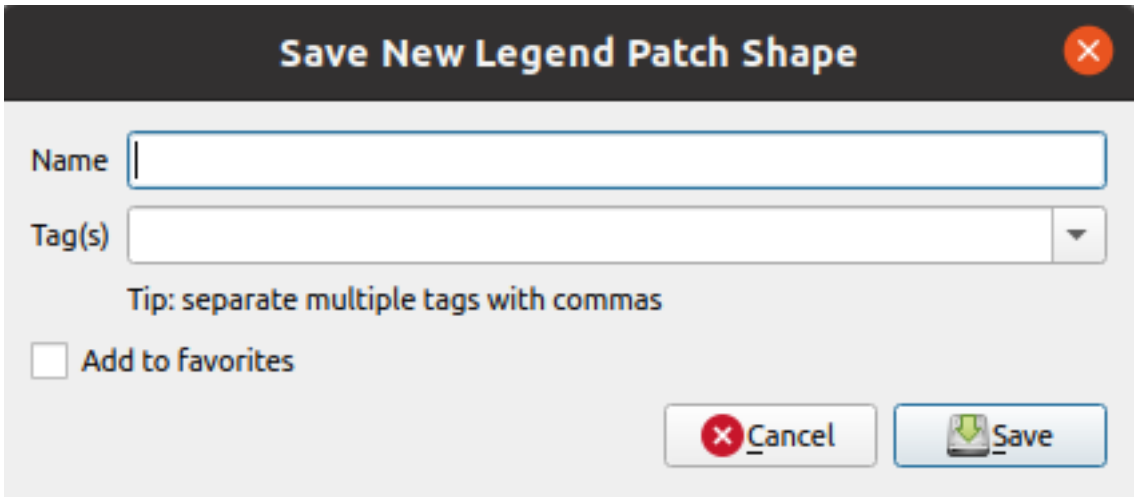


図 14.9: 新規凡例パッチを保存

ここでは名前を入力し、図形を説明するタグやお気に入りに追加するかどうかを選択する必要があります。

保存... ボタンを押すとこの図形がリストに追加され、 **新規凡例パッチ** ダイアログに戻り、新しい図形の作成を続けられます。

14.2 シンボルセレクタ

シンボルセレクタはシンボルをデザインするためのメインダイアログです。マーカー、ライン、塗りつぶしのシンボルを作成したり、編集したりすることができます。

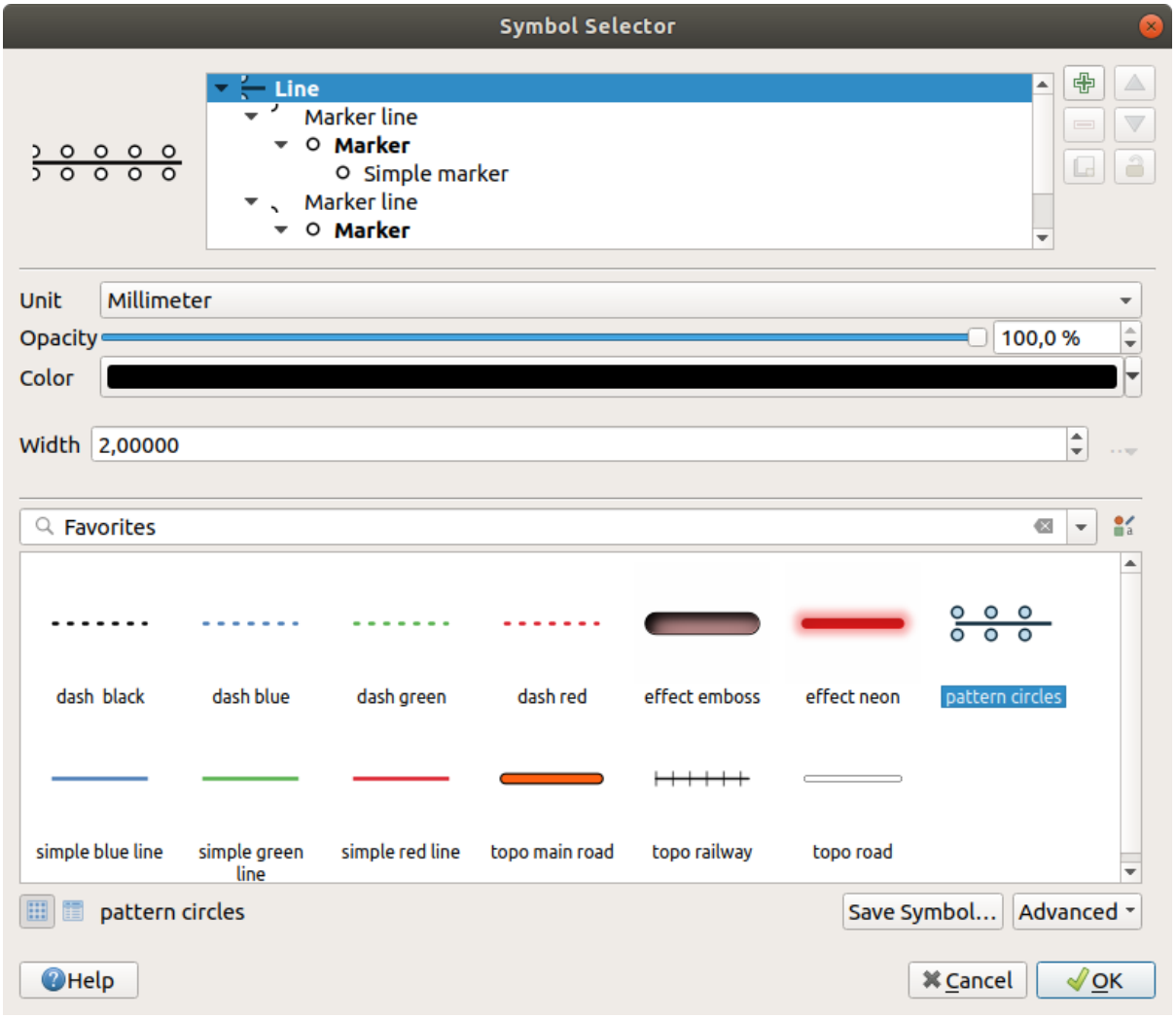


図 14.10: ラインシンボルのデザイン

シンボルセレクタダイアログは 2 つの主要な要素からなります：

- 1 つはシンボルツリーで、新たなグローバルシンボルを作るために結合されたシンボルレイヤを表示します。
- そしてもう 1 つは、ツリー内で選択されたシンボルレイヤの構成の設定です。

14.2.1 シンボルレイヤツリー

あるシンボルはいくつかの **シンボルレイヤ** から構成されます。シンボルツリーはこれらのシンボルレイヤの重なりを表示しており、これが新しいグローバルシンボルを形作るために結合されます。また、シンボルのプロパティが変更されると、動的なシンボル表現はすぐに更新されます。

シンボルツリー内で選択されたアイテムのレベルにもよりますが、ツリーの管理に役立つ様々なツールが利用可能です：

-  シンボルレイヤを追加：好きなだけシンボルレイヤを重ねることができます
-  シンボルレイヤを削除
- シンボルレイヤの色を固定：  ロックされた色は、ユーザーがグローバル（または上位の）シンボルレイヤの色を変更しても変更されません
-  シンボルレイヤ（のグループ）を複製
- シンボルレイヤを上下に移動

14.2.2 シンボルの設定

QGIS では、シンボルの設定は 2 ステップになっています：シンボルと、シンボルレイヤです。

シンボル

ツリーの最上位レベルはレイヤのジオメトリによって異なり、**マーカー**、**ライン** もしくは **塗りつぶし** タイプです。各シンボルは、1 つ以上のシンボル（ほかのタイプのシンボルも含む）またはシンボルレイヤを埋め込むことができます。

グローバルシンボルに適用するパラメータをいくつか設定することができます：

- **単位**：**ミリメートル（Millimeters）**、**ポイント（Points）**、**ピクセル（Pixels）**、**縮尺済みメートル（Meters at Scale）**、**地図単位（Map units）** もしくは **インチ（Inches）** のどれか（詳細については **単位セレクタ** を参照）
- **不透明度**
- **色**：このパラメータがユーザーによって変更されると、この値は全てのロックされていないサブシンボルの色にも適用されます
- **大きさ** と **回転**：マーカーシンボルに対する設定
- **幅**：ラインシンボルに対する設定

Tip: シンボルレベルの **大きさ**（マーカーシンボル）や、**幅**（ラインシンボル）のプロパティを使用すると、シンボルに埋め込まれた全ての **シンボルレイヤ** の寸法がこれに比例してリサイズされます。

注釈: 幅、大きさ、回転パラメータの横にある **データによって定義された上書き** ボタンは、スタイルマネージャダイアログから設定する場合には使用不可になります。シンボルが地図レイヤに結合されると、このボタンは **比例シンボルや多変量解析** レンダリングを作成するのに役立ちます。

- **シンボルライブラリ** のプレビュー：同じタイプのシンボルが表示され、すぐ上にある編集可能なドロップダウンリストから、自由形式のテキストや **カテゴリ** でフィルタリングすることができます。また、  スタイルマネージャ ボタンを使用して同名のダイアログを開き、シンボルのリストを更新することもできます。 そこでは、 **スタイルマネージャ** のセクションで公開されている機能を使用することができます。

  シンボルは以下のいずれかの方法で表示されます：

  - フレームの下にある  リストビュー ボタンを使用して、アイコンのリスト形式（サムネイル、名前、関連付けられたタグ）表示
  - または、  アイコンビュー ボタンを使用して、アイコンのプレビュー表示
- **シンボルを保存** ボタンを押すと、編集中のシンボルをシンボルライブラリに追加できます。
- **詳細設定**  オプションでは、以下の設定ができます：
  - ラインと塗りつぶしシンボルに対しては、 **キャンバス範囲に地物を切り抜く** かどうかの設定
  - 塗りつぶしシンボルに対して **右手ルールを強制** ：これにより、レンダリングされる塗りつぶし記号を、リングの向きに関する標準的な「右手ルール」（つまり、外側のリングが時計回り、内側のリングがすべて反時計回りのポリゴン）に強制的に従わせることができます。

    この方向の強制固定はレンダリング時にのみ適用され、元の地物ジオメトリは変更されません。これにより、レンダリングされるデータセットや個々の地物のリングの向きとは無関係に、一貫性のある見た目の塗りつぶしシンボルが作成されます。
  - シンボルに適用されているレイヤの **シンボロジ** に応じて、 **アドバンスト** メニューでは以下の追加設定ができます：
    * **描画順序...** ：シンボルのレンダリング順序の定義
    * **データによる凡例サイズの定義**
    * **保存されたシンボルに一致** と **ファイルからのシンボルに一致...** で自動的に **シンボルを分類に割り当て**

シンボルレイヤ

ツリーの下位レベルでは、シンボルレイヤをカスタマイズすることができます。利用可能なシンボルレイヤのタイプは、上位のシンボルタイプに依存します。シンボルレイヤに  **描画エフェクト** を適用し、レンダリングを拡張することができます。

全てのシンボルレイヤタイプのオプションについて述べるのは不可能なので、以下では特別で重要なものに限って述べています。

共通のパラメータ

マーカー、ライン、塗りつぶしサブタイプにかかわらず、いくつかの共通のオプションとウィジェットがシンボルレイヤを作成するために利用できます。

- 色操作を簡単に操作するための **色セレクタ** ウィジェット
- **単位**：**ミリメートル（Millimeters）**、**ポイント（Points）**、**ピクセル（Pixels）**、**縮尺済みメートル（Meters at Scale）**、**地図単位（Map units）** もしくは **インチ（Inches）** のどれか（詳細については **単位セレクタ** を参照）
- ほとんど全てのオプションの横にある  データによって定義された上書き ウィジェット：各シンボルのカスタマイズ機能を拡張（詳細は **データによって定義された上書きの設定** を参照）
-  **シンボルレイヤを有効化** オプションは、シンボルレイヤの可視性をコントロールします。無効化されたシンボルレイヤはレンダリング時に描画されませんが、シンボル内には保存されています。シンボルレイヤを非表示にできるようになることで、シンボルの最適なデザインを探す際にテストでシンボルレイヤを削除する必要がないため、便利になります。データ定義の上書きはこれに加え、様々なシンボルレイヤを式（すなわち地物属性）に基づいて表示・非表示にすることができます。
-  **描画エフェクト** ボタン： **レンダリング効果** を設定する際に使用

注釈: 以下の説明では、シンボルレイヤタイプが地物のジオメトリにバインドされていることを前提としていますが、シンボルレイヤはお互いに埋め込むことができることに留意してください。この場合、下位レベルのシンボルレイヤのパラメータ（配置、オフセットなど...）は、地物ジオメトリ自体ではなく、上位レベルのシンボルにバインドされているかもしれません。

マーカーシンボル

マーカーシンボルはポイントジオメトリの地物に適したシンボルで、いくつかの **シンボルレイヤタイプ** があります：

- **シンプルマーカー**（デフォルト）

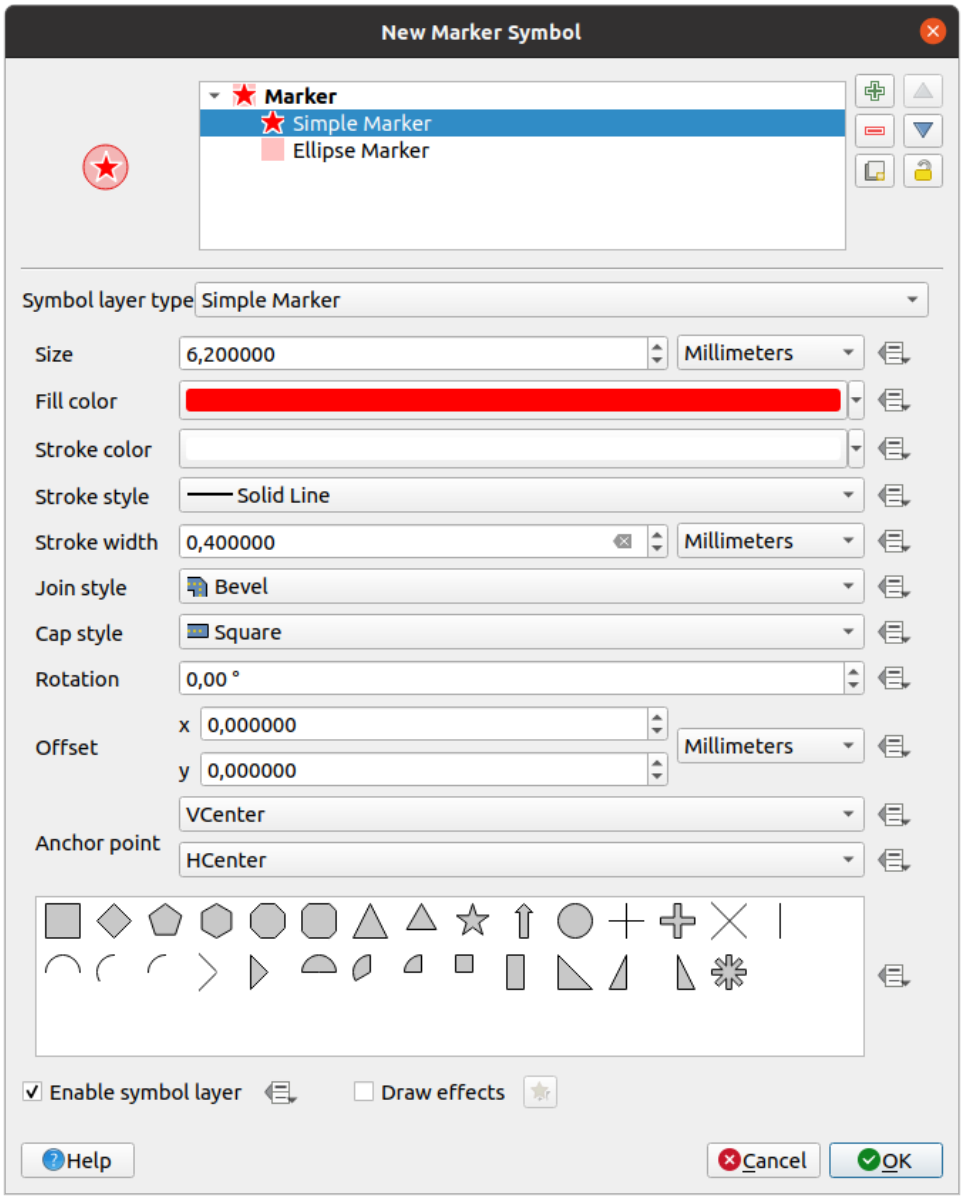


図 14.11: シンプルマーカーのシンボルをデザインする

シンプルマーカーシンボルレイヤタイプには、以下のプロパティがあります:

- **大きさ** ：さまざまな単位をサポートします
- **塗りつぶし色**
- **ストローク色** 、定義済みリストからの **ストロークスタイル** 、 **ストローク幅**
- **継ぎ目スタイル** ： **Bevel** 、 **Miter** 、 **Round** から選択します
- **両端スタイル** ： **Square** 、 **Flat** 、 **Round** から選択します
- **回転**
- 地物からの *X* および *Y* 方向の **オフセット**
- **アンカーポイント** ：シンボル上で象限点を定義し、配置の原点とします。これは、 **オフセット**

が適用されるポイントです。

- **楕円マーカー**：シンプルなマーカーシンボルのレイヤで、シンボル幅と高さがカスタマイズ可能です
- **塗りつぶしマーカー**：**塗りつぶしのサブシンボル** をマーカーのレンダリングに使用する点を除けば、シンプルマーカーのシンボルレイヤと似ています。塗りつぶしのサブシンボルは、マーカーのレンダリングに既存の QGIS の塗りつぶし（とストローク）スタイル、例えばグラデーションや Shapeburst 塗りつぶしを使用することができます。
- **フォントマーカー**：シンプルマーカーのシンボルレイヤと似ていますが、インストールされたフォントをマーカーとして使用してレンダリングする点が異なります。追加プロパティには以下のものがあります：
  - **フォントファミリー**
  - **フォントスタイル**
  - **文字** は、シンボルとして表示される文字（列）です。文字は入力するか、フォント文字コレクションウィジェットから選択することができ、選択した設定はライブ **プレビュー** で確認できます。
- **ジオメトリジェネレータ**（**ジオメトリジェネレータ** を参照）
- **マスクグリッド（Mask）**：このシンボルのサブシンボルはマスク形状を定義し、色の特性は無視されて不透明度だけが使用されます。これは、マーカーシンボルが色の近いラベルや他のシンボルと重なって見えづらい場合に便利です。詳細は **マスクプロパティ** を参照してください。
- **ラスタ画像マーカー**：マーカーシンボルとして画像（`PNG`、`JPG`、`BMP` 等）を使用します。画像はディスク上のファイル、リモート URL、スタイルデータベースに埋め込まれたもの（**詳細はこちら**）、または base64 文字列が使用できます。画像の幅と高さは独立に設定することもできますし、 

ことも できます。サイズは任意の **共通の単位** を使用するか、画像のオリジナルサイズに対するパーセント（幅基準でスケール）で設定できます。
- **ベクタフィールドマーカー**（**ベクタフィールドマーカー** 参照）
- **SVG マーカー**：SVG パス（**設定** **オプション...** **システム** メニューで設定されたパス）にある画像をマーカーシンボルとしてレンダリングできます。シンボルの幅と高さは独立に設定することもできますし、 

ことも できます。各 SVG ファイルの色やストロークを調整することもできます。画像はディスク上のファイル、リモート URL、スタイルデータベースに埋め込まれたもの（**詳細はこちら**）、または base64 文字列が使用できます。

  シンボルは、*Dynamic SVG parameters* で設定することもできます。SVG シンボルのパラメータ化については、**パラメータ付き** *SVG* のセクションを参照してください。

注釈: SVG のバージョン要件

QGIS は SVG Tiny 1.2 プロファイル に従った SVG ファイルをレンダリングします。このプロファイルは、携帯電話や PDA からノートパソコンやデスクトップコンピュータまで、さまざまなデバイスへの実装が想定されているため、SVG 1.1 Full に含まれる機能のサブセットに加えて、SVG の機能を拡張する新しい機能も含まれています。

この仕様に含まれていない機能は、QGIS では正しくレンダリングされないものがあるかもしれません。

ラインシンボル

ラインシンボルはラインジオメトリの地物に適したシンボルで、以下のようなシンボルレイヤタイプがあります：

- **直線（Simple Line）**（デフォルト）

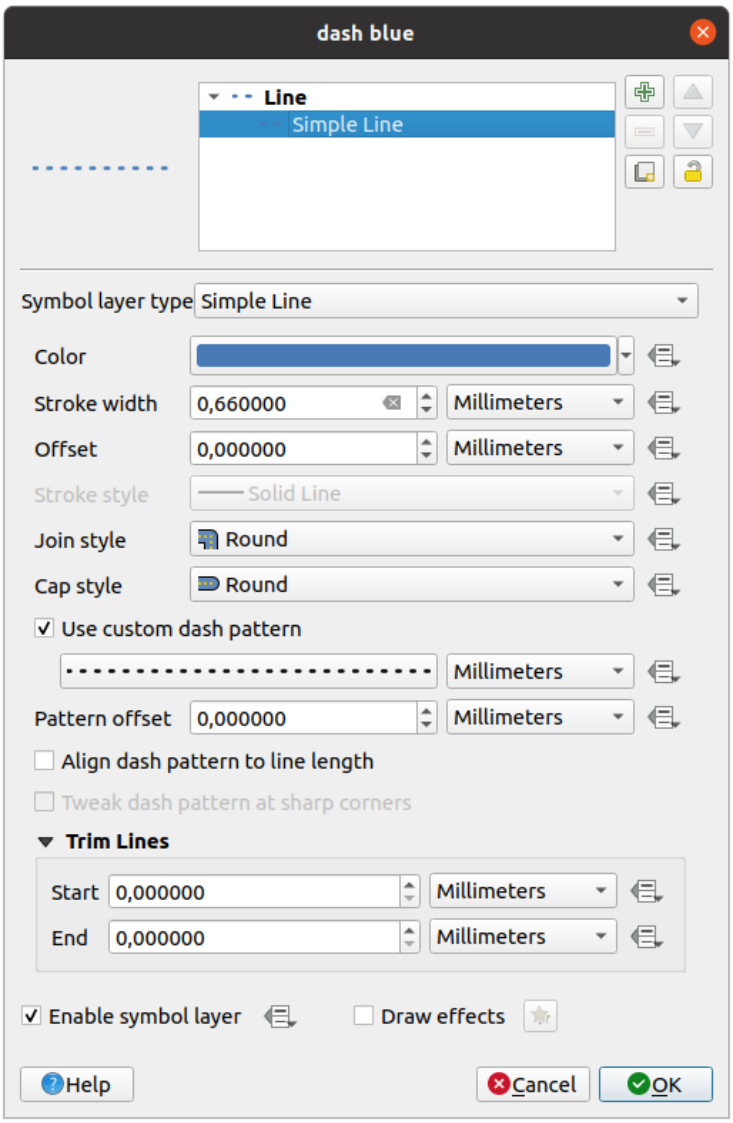


図 14.12: シンプルラインのシンボルをデザインする

シンプルラインのシンボルレイヤタイプは **シンプルマーカーシンボル** と同様のプロパティが多数ありますが、これに加えて、

- **破線を使用** ： **ストロークスタイル** の設定をカスタムの破線で上書きします。破線のモデルを形作る、連続した破線（dash）とスペースの長さを選択した単位で定義する必要があります。

パターンの長さの合計がダイアログの下部に表示されます。

- **パターンオフセット** ：ラインにおける破線と空白の位置を微調整して、ラインの角を考慮したより良い位置に破線を配置できます（また、これは隣接する破線パターンの境界を「揃える」ために使用することもできます）。
- **破線の間隔を線の長さに調整する** ：破線パターンの長さを調整し、線の終わりが空白ではなく完全な破線要素で終わるようにします。
- **破線の間隔を鋭角に調整する** ：破線パターンの配置を動的に調整して、破線要素が鋭角部に出入りし、鋭角が完全な破線要素で表現されるようにします。これは **破線の間隔を線の長さに調整する** の設定に依存します。
- **開始** や **終了** 位置からの **線をトリミング** ：これにより、ラインを描画する際に実際のラインストリングから最初の x mm と最後の y mm を切り取ったラインをレンダリングできます。これはライン全体の長さに対するパーセンテージも含め様々な **単位** をサポートしており、さらに細かく制御するためにデータ定義とすることもできます。開始・終了のトリミング距離は例えば、ラインの始点と終点に配置されたマーカーシンボルレイヤにラインレイヤがに重なってはいけないような、複雑なシンボルを作成する場合に使用します。

- **矢印** ：ラインを曲がった（または直線的な）矢印として描画します。ヘッドは一方向または両方向とすることができ、ヘッドの設定が可能です（データ定義とすることもできます）。
  - **ヘッドの種類**
  - **矢印の種類**
  - **矢印の幅**
  - **矢印の始点での幅**
  - **ヘッドの長さ**
  - **ヘッドの太さ**
  - **オフセット**

  湾曲した矢印（ライン地物は少なくとも 3 つの頂点を持っていなければなりません）を作成したり、 **各セグメント上に矢印を繰り返す** ことができます。また、矢印のボディのレンダリングにはグラデーションや shapeburst といった **塗りつぶしシンボル** を使用します。ジオメトリジェネレータと組み合わせると、このタイプのレイヤシンボルはフローマップを表現するのに役立ちます。

- **ジオメトリジェネレータ** （ **ジオメトリジェネレータ** を参照）
- **補間された線** ：ラインのレンダリングについて、ラインの **ストローク幅** や **色** を一定（パラメータを **固定幅** や **単一色** に設定）としたり、ジオメトリに沿って変化させたりすることができます。可変とする場合には、以下の入力が必要です：
  - **開始値** と **終了値** ：地物ジオメトリの両端における値で、値の補間のために使用します。この値は、固定値、地物の属性、または式に基づく値を使用できます。
  - **最小値** と **最大値** ：補間が実行される値の上下限です。 Load ボタンを押すと、レイヤに適用された開始値・終了値の最小値と最大値に基づいて自動的に値が入力されます。
  - ストローク幅オプションにのみ、以下の設定があります：

    * **最小幅** と **最大幅** ：可変幅の範囲を定義します。 **最小値** には **最小幅** が、 **最大値** には **最大幅** が割り当てられます。幅の値には **単位** を関連付けられます。
    * **絶対値を使用** ：補間には絶対値を考慮します（負の値は正の値として使用します）。
    * **範囲外の値を無視** ：地物の [開始値 - 終了値] の範囲が [最小値 - 最大値] の範囲に含まれない場合、デフォルトでは範囲外の部分の地物ジオメトリは最小幅や最大幅でレンダリングされます。このオプションにチェックを入れると、範囲外の部分の地物ジオメトリは全くレンダリングされません。
  - 可変色については、 **カラーランプ** の任意の補間方法を利用できます。

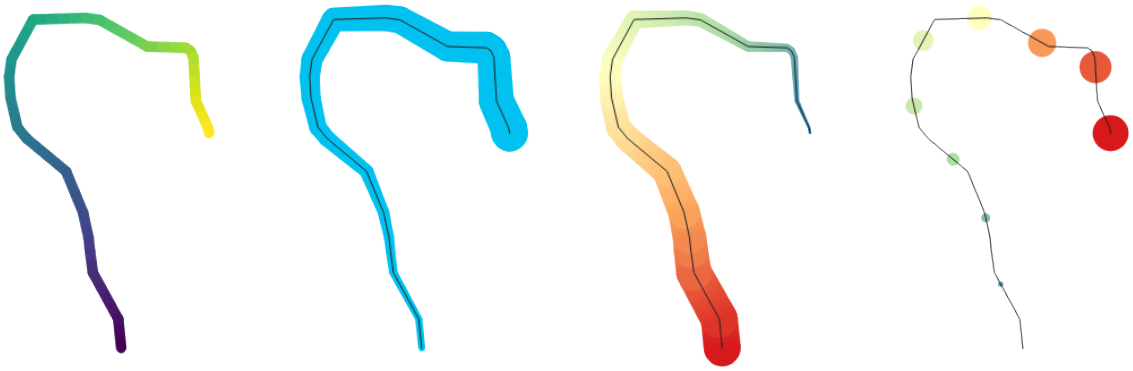


図 14.13: 補間された線の例

- **マーカーライン** ：ラインの長さにわたって **マーカーシンボル** を繰り返します。
  - **マーカーの位置** ：位置は、（最初の頂点から開始する）等間隔の設定にすることもできますが、ラインジオメトリのプロパティに基づく設定（最初の点、最後の点、内部の点、ラインの中央の点、セグメントの中央の点、すべてのカーブ点）にすることもできます。
  - 最初の点または最後の点を有効にすると、 **各パートの両端に配置** オプションを使用でき、マルチパートジオメトリの各パートの最初または最後の頂点にもマーカーをレンダリングします。
  - **ラインに沿ったオフセット** ：マーカーの配置には、好みの ref:**単位** <*unit_selector*> （ミリメートル、ポイント、地図単位、縮尺済みメートル、パーセントなど）で指定した、ラインに沿った方向のオフセットを指定することもできます。
    * 正の値を指定すると、（マーカーの位置が **最初の点** や *With interval* の場合には）ラインの方向にマーカーシンボルをオフセットします。位置が **最後の点** の場合は反対方向にオフセットします。
    * 負の値を指定すると、ラインが閉じていない場合には、オフセットは生じない（位置が **最初の点** や **最後の点** の場合）か、（最後の頂点から）反対向きにシンボルをオフセットします。
    * 閉じたリングの場合には、QGIS はリングでオフセットが（前方または後方に）周回するものとして扱います。例えば、オフセットを 150%（または -10%、-110%）に設定すると、オフセットは閉じたリングの 50%（またはいずれも 90%）として扱います。
  - **線の向きに沿って回転** オプションは、各マーカーシンボルをラインの方向に対する相対的な向きとするかどうかを設定します。

ラインはさまざまな方向のセグメントの連続であることが多いので、マーカーの回転は線に沿って指定された距離を平均することで計算されます。例えば、 **角度の平均化範囲** プロパティを 4mm に設定すると、シンボル位置の前後 2mm にある、線に沿った 2 点がそのマーカーシンボルの線の角度を計算するのに使われます。これは全体的な線の方向からの局所的なわずかな変動を平滑化（あるいは除去）する効果があり、結果としてマーカーラインシンボルの見た目の方向がより良くなります。

  - **オフセット量** ：マーカーシンボルをライン地物からオフセットさせることもできます。

- **ハッシュ線** ：ラインシンボルの長さにわたって線分（ハッシュ）を繰り返します。ハッシュはラインのサブシンボルを使用して個々のセグメントがレンダリングされます。言い換えると、ハッシュ線はマーカーシンボルが線分に置き換わったマーカーラインのようなものです。従って、ハッシュ線はマーカーラインと **同様のプロパティ** を持ち、これに加えて以下のプロパティがあります：

  - **ハッシュ線の長さ**
  - **ハッシュ線の回転**

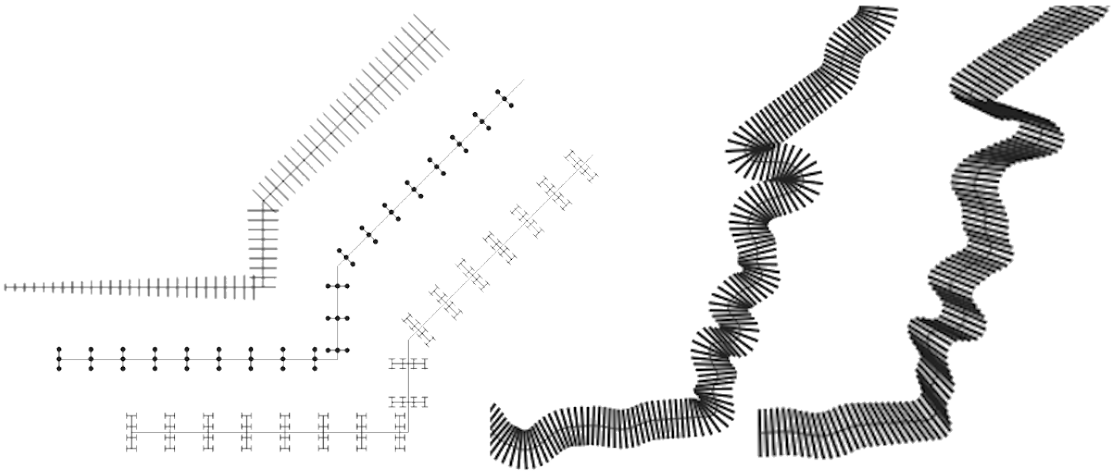


図 14.14: ハッシュ線の例

- **ラスタ線**: ライン地物の形状の長さに従ってラスタ画像をレンダリングし、繰り返します。guilabel: **ストローク幅**, **オフセット**, **継ぎ目スタイル**, **両端スタイル**, **不透明度** を調整することができます。

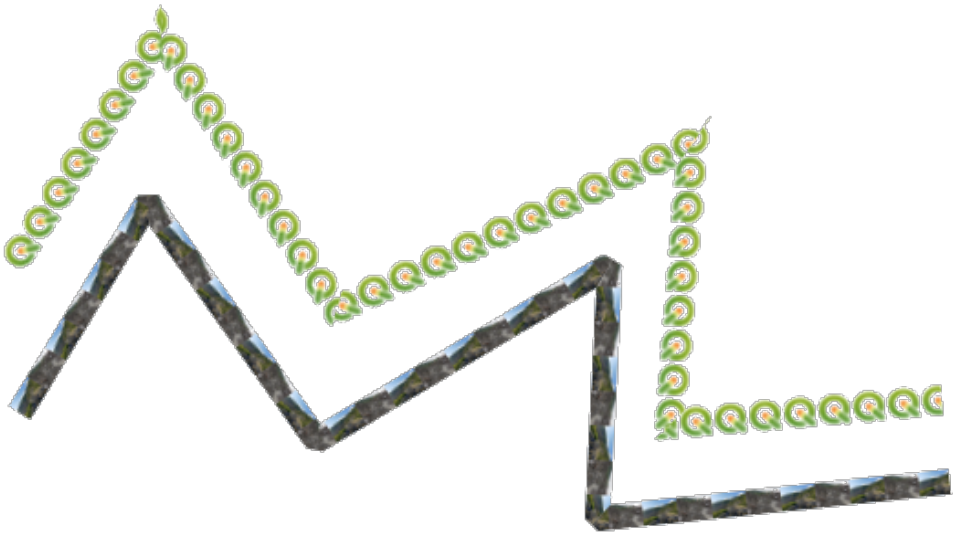


図 14.15: ラスタ線の例

- **線バースト**: 線の幅に沿ってグラデーションを描画します。*2* **つの色** または **カラーランプ** を選択することができ、**ストローク幅**, **オフセット**, **継ぎ目スタイル**, **両端スタイル** を調整することができます。

図 14.16: 線バーストの例

塗りつぶしシンボル

ポリゴンジオメトリの地物に適するように、塗りつぶしシンボルにも以下のような様々なシンボルレイヤタイプがあります：

- **シンプル塗りつぶし**（デフォルト）：ポリゴンを均一な色で塗りつぶします

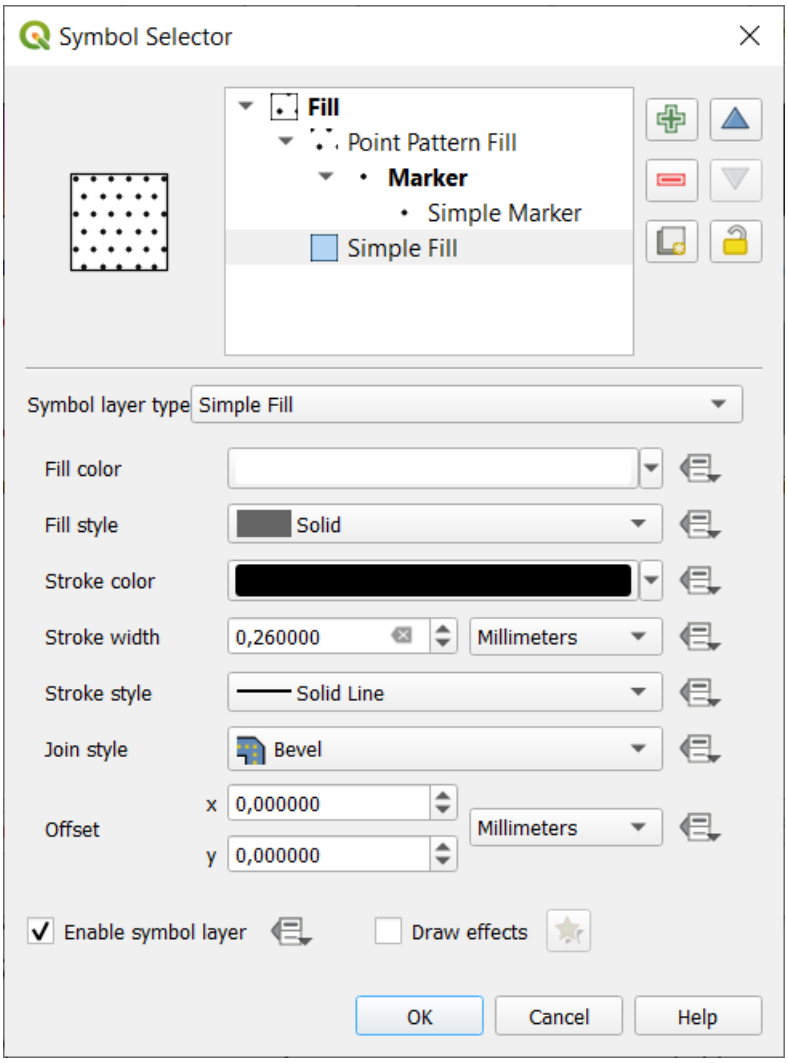


図 14.17: シンプル塗りつぶしのシンボルをデザインする

- **重心塗りつぶし**：表示されている地物の重心位置に **マーカーシンボル** を配置します。重心位置の計算はマップキャンバスで表示されている領域にクリップされたポリゴンに対して行われ、穴は無視されているため、マーカーの位置は地物の実際の重心位置ではないかもしれません。正確な重心位置が必要な場合には、 **ジオメトリジェネレータシンボル** を使用してください。

  以下の設定ができます：

  - **マーカーをポリゴンの内側に強制移動する**
  - **マルチパート地物の全パートにマーカーを描画** するか、最大パートにのみ描画するかを設定できます
  - マーカーシンボルは全体を表示するか、それとも一部を表示するかを設定できます。一部表示

は、現在の地物ジオメトリと重複する部分を残す（**ポリゴンでマーカーを切り抜く**）か、シンボルが属するジオメトリパートと重複する部分を残す（**現在の地物の部分でマーカーを切り抜く**）かを設定できます。

- **ジオメトリジェネレータ**（**ジオメトリジェネレータ** を参照）
- **グラデーション塗りつぶし**: 単純な 2 色のグラデーション、または事前に定義された **グラデーションカラーランプ** に基づく、放射状、線形、または円錐状のグラデーションを使用してポリゴンを塗りつぶします。グラデーションは回転させることができ、1 つの地物ごとに適用することも、マップの範囲全体に適用することもできます。また、始点と終点は座標または（地物またはマップの）重心を使用して設定できます。データ定義のオフセットも定義できます。
- **ラインパターン塗りつぶし**：ポリゴンを **ラインシンボルレイヤ** によるハッチングパターンで塗りつぶします。次の設定ができます：
  - **整列**: パターンと地物の位置関係を定義します：
    - **パターンを地物に揃える**: 各地物内にラインが描画されます。
    - **パターンを地図領域に揃える**: パターンがマップの範囲全体にレンダリングされ、ラインが地物を横切ってきれいに整列します
  - ラインを反時計回りに **回転** します
  - **間隔**: 2 つの隣り合うラインの距離
  - 地物の境界線からの **オフセット** 距離
  - **切り抜き**: 塗りつぶす線をどのようにポリゴンの形に切り抜くかを制御できます。オプションは以下の通りです：
    - **レンダリング中のみクリッピング**: 地物のバウンディングボックス全体を覆うラインが生成され、描画中にクリップされます。線の両端（始点と終点）は表示されません。
    - **レンダリング前に線をクリッピング**: 線はレンダリング前にポリゴンの正確な形に切り抜かれます。線の両端（両端スタイル、始点/終点マーカーラインオブジェクトなどを含む）は表示され、（ラインシンボルの設定によっては）ポリゴンの外にはみ出ることもあります。
    - **クリッピングしない**: クリッピングは全く行われません。ラインは地物のバウンディングボックス全体を覆います
- **ポイントパターン塗りつぶし**：ポリゴンを **マーカーシンボル** によるグリッドパターンで塗りつぶします。次の設定ができます：
  - **整列**: パターンと地物の位置関係を定義します：
    - **パターンを地物に揃える**: マーカーラインが各地物の中にレンダリングされます
    - **パターンを地図領域に揃える**: パターンがマップ範囲全体にレンダリングされ、マーカーが地物をまたいできれいに配置されます

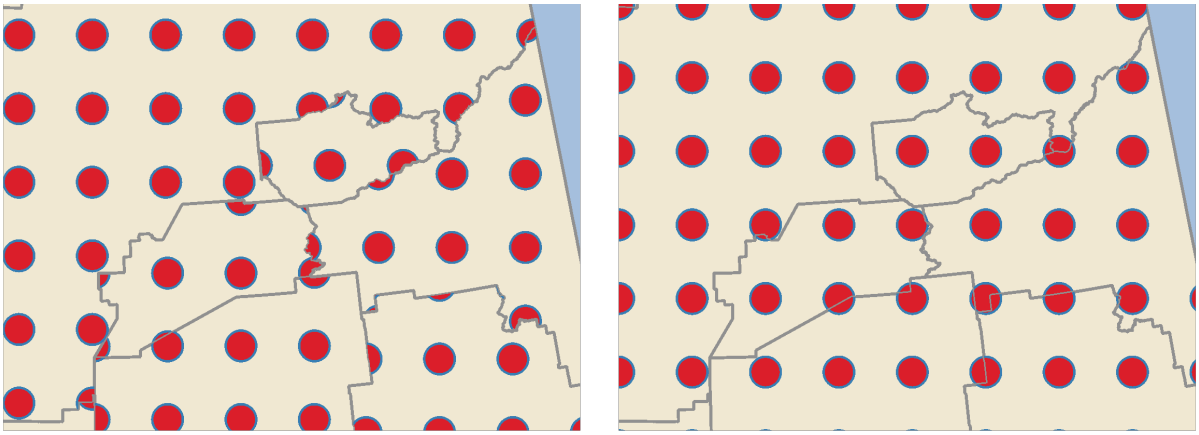


図 14.18: 地物（左）とマップ範囲（右）へのポイントパターンの整列

- **距離** ：2 つの隣り合うマーカーどうしの **水平** 距離と **垂直** 距離
- **変位** ：連続する列方向（または行方向）のマーカー間の並びの **水平** （または **垂直** ）方向のオフセット
- **オフセット** ：地物の境界線からの **水平** 距離および **垂直** 距離
- **切り抜き**: 塗りつぶすマーカーをどのようにポリゴンの形に切り抜くかを制御できます。オプションは以下の通りです：
  - **切り抜くシェープ**: ポリゴンの内側の部分だけが見えるようマーカーを切り抜きます
  - **シェープ内のマーカー重心**: マーカーの重心がポリゴンの内側にあるマーカーだけが描画されますが、これらのマーカーはポリゴンの外に出ないよう切り抜かれません
  - **シェープ内のマーカー**: ポリゴンの中に完全に入るマーカーだけが表示されます
  - **クリッピングしない**: ポリゴンと交差するマーカーは完全にレンダリングされます（厳密には「マーカーのバウンディングボックスと交差する」です）

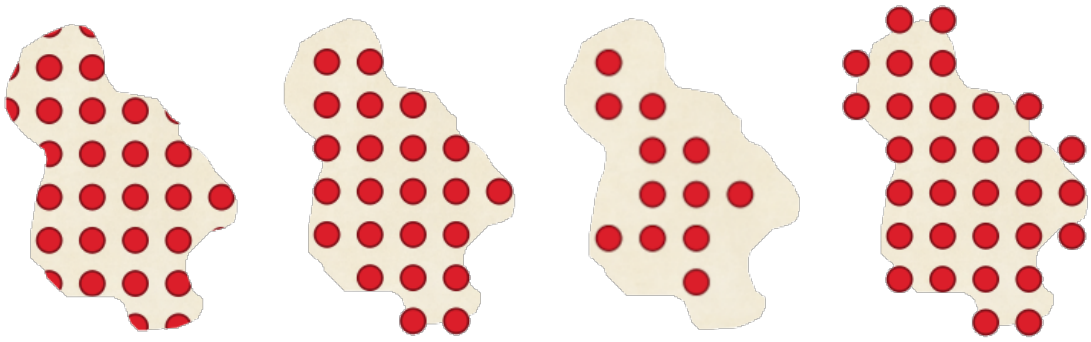


図 14.19: 塗りつぶしにおけるマーカーの切り抜き - 左から右へ： 切り抜くシェープ、シェープ内のマーカー重心、シェープ内のマーカー、クリッピングしない

- パターン全体を時計回りに **回転** します
- **パターンをランダム化** グループ設定では、ポイントパターン塗りつぶしの各ポイントを、指定した最大距離 **水平方向** または **垂直方向** までランダムにずらすことができます。最大オフセッ

トは、ミリメートル、ポイント、地図単位、あるいは「パーセント」(パーセントはパターンの幅や高さに対する相対値) など、サポートされている任意の単位で指定できます。

オプションで乱数シードを設定することで、マップの更新時にシンボルパターンが「飛び跳ねる」のを防ぐことができます。データによる定義の上書きにも対応しています。

注釈: パターンをランダム化 と **ランダム塗りつぶし** シンボルタイプの主な違いは、ポイントパターンによるランダムオフセットでは、マーカーを準「規則的」に配置できることです — パターン内の点が効果的にグリッドに拘束されるため、空の領域やマーカーの重なりがないセミランダムな塗りつぶしを作成できます。(ランダム塗りつぶしとは対照的に、ポイントは常に完全にランダムに配置されます... その結果、点の視覚的なクラスタや不要な空白領域が生じることがあります)。

- **ランダム塗りつぶし** ：ポリゴン境界内のランダムな位置に配置した **マーカーシンボル** でポリゴンを塗りつぶします。以下の設定ができます：
  - **カウント方法** ：レンダリングするマーカーシンボルの数について、絶対値カウントとするか、または密度ベースのカウントとするか
  - **点の数** ：レンダリングするマーカーシンボルの数
  - オプションの乱数 **シード** を設定することで、一貫性のある配置にできます
  - **面積密度** ：密度ベースのカウント方法の場合は、マップが更新されるたびに、異なるスケールやズームレベルでもマーカーの塗りつぶし密度が同じままとなります（QGIS サーバーやタイルベースのレンダリングでもうまく配置されるようなランダム配置も可能です）
  - **ポリゴンでマーカーを切り抜く** ：端付近にレンダリングされたマーカーをポリゴン境界で切り抜くかどうか
- **ラスタイメージ塗りつぶし** ：ポリゴンをラスタ画像（ PNG 、 JPG 、 BMP 等）でタイル状に塗りつぶします。画像はディスク上のファイルや、リモート URL、文字列でエンコードされた埋め込みファイル（ **詳細はこちら** ）が利用できます。オプション（データ定義可）には、不透明度、画像の幅、座標モード（オブジェクトまたはビューポート）、回転およびオフセットがあります。画像の幅は任意の **共通の単位** を使用するか、画像のオリジナルサイズに対するパーセントで設定できます。
- **SVG 塗りつぶし**: 与えられた大きさ（**テクスチャ幅**）の SVG マーカー を用いてポリゴンを塗りつぶします。
- **Shapeburst 塗りつぶし**: ポリゴンの境界からポリゴンの中心に向かってグラデーションが描かれるグラデーション塗りつぶしをバッファリングします。設定可能なパラメータには、境界からシェードまでの距離、カラーランプまたは単純な 2 つの色のグラデーションの使用、オプションで塗りつぶしのぼかし、オフセットなどがあります。
- **アウトライン：矢印** ：ポリゴン境界の表現にラインの **矢印シンボル** レイヤを使用します。アウトライン矢印の設定は、ラインの矢印シンボルの設定と同様です。
- **アウトライン：ハッシュ線** ：ポリゴン境界（ **リング** ）の表現に **ハッシュ線シンボル** レイヤを使用します。リングは、内側リングのみ、外側リングのみ、またはすべてのリングを対象にできます。その他の設定は、ラインのハッシュ線シンボルと同様です。

- **アウトライン：マーカーライン** ：ポリゴン境界（ **リング** ）の表現に **マーカーラインシンボル** レイヤを使用します。リングは、内側リングのみ、外側リングのみ、またはすべてのリングを対象にできます。その他の設定は、ラインのマーカーラインシンボルと同様です。
- **アウトライン：直線（Simple line）** ：ポリゴン境界（ **リング** ）の表現に **直線シンボル** レイヤを使用します。リングは、内側リングのみ、外側リングのみ、またはすべてのリングを対象にできます。**ポリゴンの内部にだけ線を描画する** オプションは、ポリゴンの内側にポリゴン境界線を表示することで、隣接するポリゴンの境界線をわかりやすく表現するのに便利です。その他の設定は、ラインシンボルと同様です。

注釈: ジオメトリタイプがポリゴンの場合、ライン/ポリゴンのキャンバス範囲への自動クリッピングを無効にするよう選択することができます。いくつかのケース（例：中心点塗りつぶしで、中心が常に実際の地物の中心でなければならない場合など）では、クリッピングによって好ましくないシンボロジが発生してしまうことがあります。

パラメータ付き SVG

SVG **マーカー** の色の変更ができます。そのためには、塗りつぶし色には `param(fill)` 、ストローク色には `param(outline)` 、ストローク幅には `param(outline-width)` というプレースホルダを追加する必要があります。これらのプレースホルダの後には、オプションでデフォルト値を指定できます。例：

```
<svg width="100%" height="100%">
<rect fill="param(fill) #ff0000" stroke="param(outline) #00ff00" stroke-width=
↪"param(outline-width) 10" width="100" height="100">
</rect>
</svg>
```

より一般的には、SVG では `param(param_name)` を使用して自由にパラメータを設定することができます。このパラメータは属性値としても、ノードテキストとしても利用できます：

```
<g stroke-width=".265" text-anchor="middle" alignment-baseline="param(align)">
  <text x="98" y="147.5" font-size="6px">param(text1)</text>
  <text x="98" y="156.3" font-size="4.5px">param(text2)</text>
</g>
```

このパラメータは、 *Dynamic SVG parameters* テーブル内で式として定義できます。

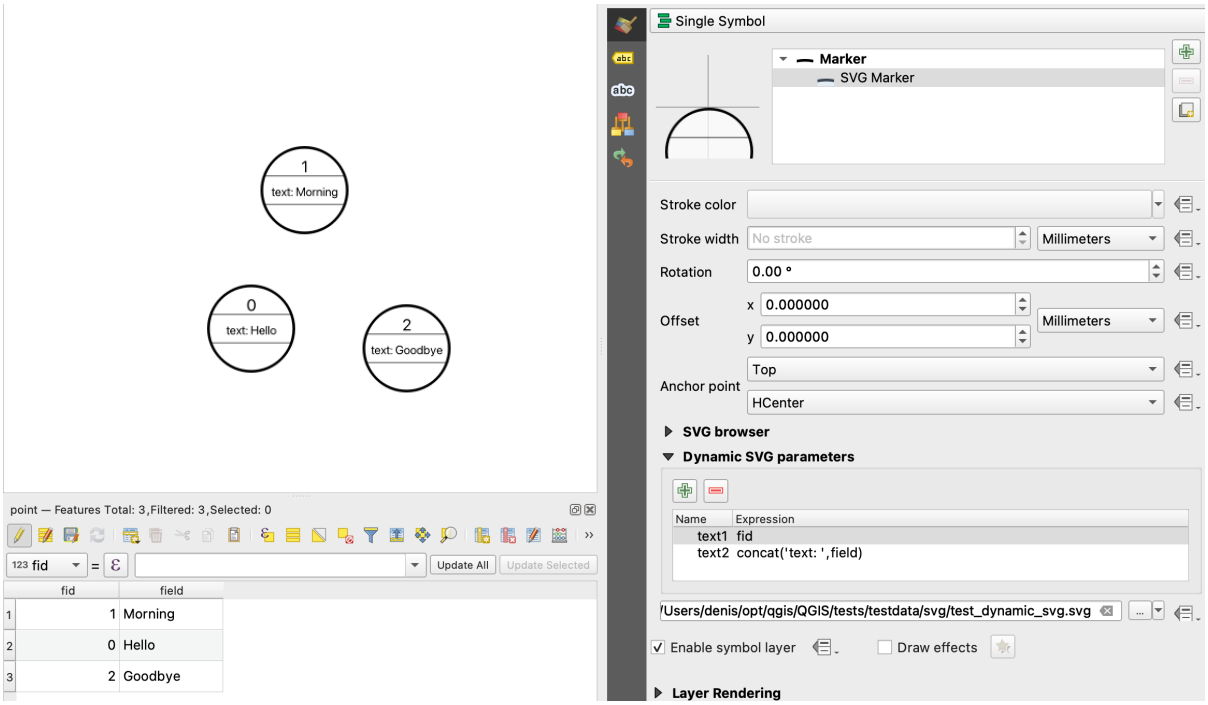


図 14.20: Dynamic SVG parameters テーブル

ジオメトリジェネレータ

すべてのタイプのシンボルで利用可能な **ジオメトリジェネレータ** シンボルレイヤは、 **式構文** を使用してレンダリング処理中にオンザフライでジオメトリを生成することができます。生成されたジオメトリは元の **ジオメトリタイプ** と一致している必要はなく、別に作成されたさまざまなシンボルレイヤを追加してお互いに重ねられます。

単位 プロパティを設定できます。ジオメトリジェネレータシンボルがレイヤに適用されるのではない場合（例えばレイアウトアイテムで使用する場合）、このプロパティによって、生成される出力結果をより制御しやすくなります。

いくつかの例：

```
-- render the centroid of a feature
centroid( $geometry )

-- visually overlap features within a 100 map units distance from a point
-- feature, i.e generate a 100m buffer around the point
buffer( $geometry, 100 )

-- Given polygon layer1( id1, layer2_id, ...) and layer2( id2, fieldn...)
-- render layer1 with a line joining centroids of both where layer2_id = id2
make_line( centroid( $geometry ),
           centroid( geometry( get_feature( 'layer2', 'id2', attribute(
               $currentfeature, 'layer2_id') ) )
         )
```

(次のページに続く)

(前のページからの続き)

```
-- Create a nice radial effect of points surrounding the central feature
-- point when used as a MultiPoint geometry generator
collect_geometries(
  array_foreach(
    generate_series( 0, 330, 30 ),
      project( $geometry, .2, radians( @element ) )
  )
)
```

ベクタフィールドマーカー

ベクタフィールドマーカーは、地殻変動や潮汐流などのベクトル場データを表示するために使用されます。これはベクトル量を、データ点の選択された属性に応じてスケーリングされ方向付けられた線（可能ならば矢印）として表示します。これはポイントデータの描画にのみ使用でき、ラインやポリゴンレイヤはこのシンボロジでは描画できません。

ベクトル場はデータの属性によって定義され、 以下のどれかで表されます：

- **デカルト座標** 成分（フィールドの x と y 成分）
- または **極座標** 系：この場合には、属性は 長さ と 角度 を定義します。角度は北から時計回りに測るか、東から反時計回りに測るかを選べ、角度単位は度もしくはラジアンです。
- **高さのみ** データ：これはデータの属性値を使ってスケーリングされた垂直方向の矢印を表示します。これは、例えば変形の鉛直成分を表現するのに適しています。

フィールドの大きさは、フィールドを見るのに適したサイズに拡大・縮小することができます。

14.3 ラベルの設定

ラベルは、ベクタ地物や地図に表示できるテキスト情報です。 ラベルは、シンボルのみでは表すことができない詳細情報を追加します。QGIS では、テキストに関連した 2 種類のアイテムが利用可能です。

- **テキスト形式** ： **フォント**、**大きさ**、**色** 、 **影** 、 **背景** 、 **バッファ** などのテキストの見た目を定義します。

  これらはマップ上のテキスト（レイアウト/マップタイトル、地図整飾、スケールバー、...）をレンダリングするのに使用され、通常は **フォント** ウィジェットを通して設定できます。

  テキスト形式 アイテムを作成するには、

  1. **スタイルマネージャ** ダイアログを開きます。
  2. **テキスト形式** タブをアクティブにします。

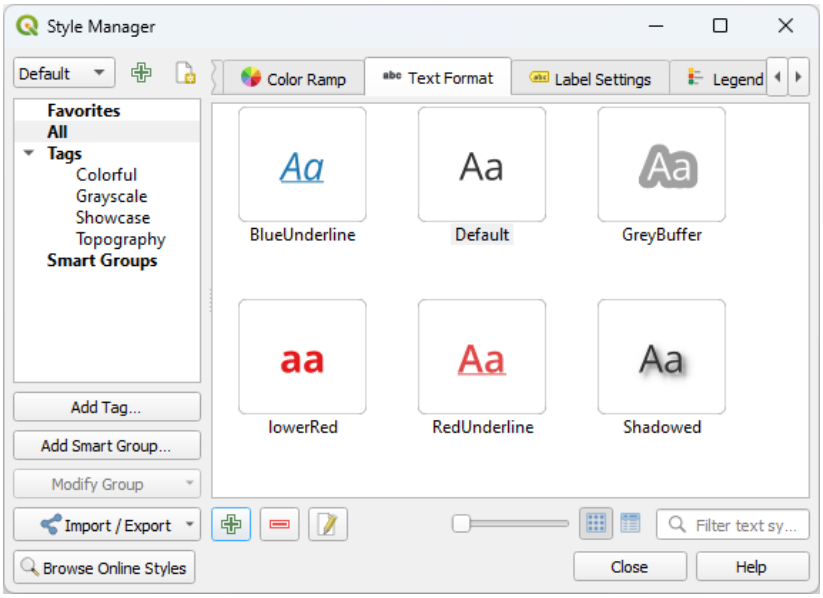


図 14.21: スタイルマネージャダイアログのテキスト形式

3. アイテム追加 ボタンを押します。 **テキスト設定** ダイアログが開き、 **設定** ができます。通常、これらのプロパティは **データ定義による上書きが可能** です。

- **ラベルの設定** ：テキストの位置や、他のテキストや地物との相互関係に関連したプロパティでテキスト形式の設定を拡張します（ **引出し線付きラベル** 、 **配置** 、 **重なり、縮尺に応じた表示** 、マスクグリッド ...）

  これらはベクタレイヤのスマートラベリングの設定に使用され、 ベクタレイヤの **レイヤプロパティ** ダイアログの **ラベル** タブまたは **レイヤスタイル** パネル、もしくは **ラベルツールバー** の レイヤラベリングオプション ボタンを使用して設定できます。

  ラベルの設定 アイテムを作成するには、

  1. **スタイルマネージャ** ダイアログを開きます。
  2. **ラベルの設定** タブをアクティブにします。

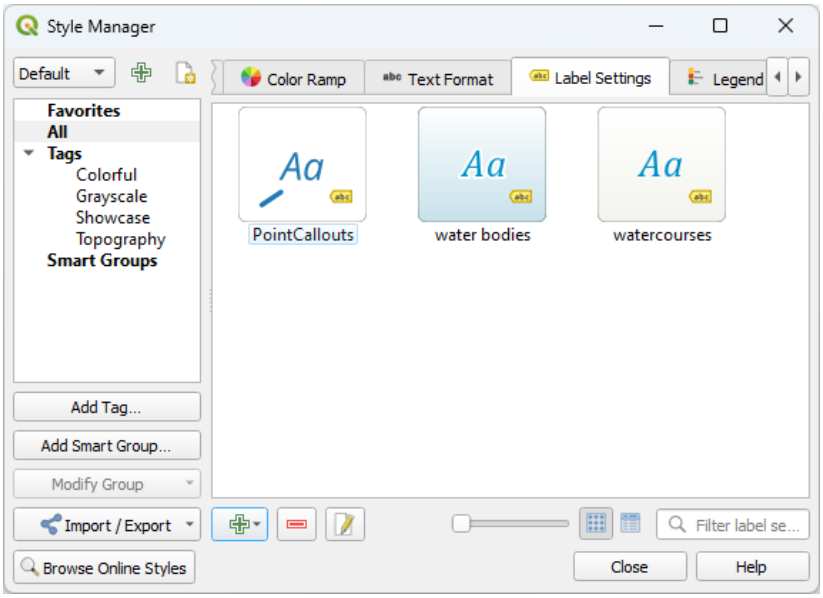


図 14.22: スタイルマネージャダイアログのラベルの設定

3. アイテム追加 メニューを押し、ラベル付けしたい地物のジオメトリタイプに応じたエントリを選択します。

ラベルの設定 ダイアログが開き、次のプロパティが表示されます。通常、これらのプロパティは **データ定義による上書きが可能** です。

14.3.1 ラベルテキストの書式設定

テキスト形式 の設定と **ラベルの設定** のいずれを設定する場合でも、以下のオプションがあります：

プロパティタブ	テキスト形式	ラベルの設定
テキスト	☑	☑
フォーマット	☑	☑
バッファ	☑	☑
マスクグリッド		☑
背景	☑	☑
影	☑	☑
引出し線付きラベル		☑
配置		☑
レンダリング		☑

テキスト

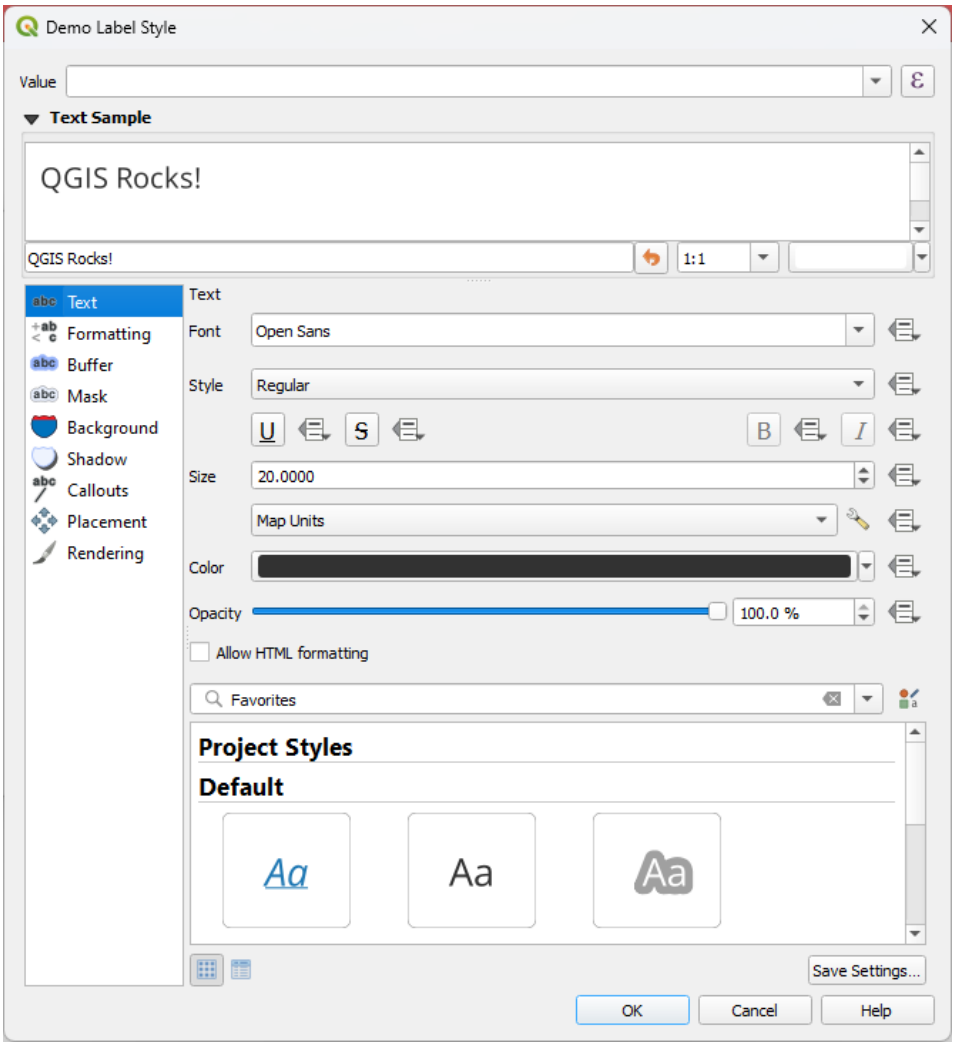


図 14.23: ラベルの設定 - テキストタブ

abc **テキスト** タブでは、次の設定を行うことができます：

- **フォント** ：マシンで使用可能なフォントに設定できます
- **スタイル**:フォントの一般的なスタイルおよび、テキストの下線または取り消し線の設定ができます
- **サイズ** は **サポートされている単位** のいずれかで設定します
- **色**
- **不透明度**
- そして *HTML* **フォーマットを使う** ： HTML フォーマットオプションはラベルをカスタマイズするためにいくつかの HTML タグを適切にレンダリングすることを可能にします。サポートされている HTML タグは以下の通りです：
  - テキスト、下線、取り消し線、上線に適用される色
  - フォントのプロパティ（フォントファミリ、フォントサイズ、太字、斜体）

HTML フォーマットを使用するには、 **値（*value*）** フィールドに HTML コードを入力する必要があります。式はパースされ、サポートされている任意の HTML タグはラベルプロパティの対応する設定を上書きします。また、HTML タグは背景、影、バッファ等のその他のラベルプロパティと組み合わせて利用できます。

以下は、HTML ベースの式とレンダリングの例です（同じラベルに異なる色と下線を適用しています）。

```
format(
  '<span style="color:blue">%1</span> ( <span style="color:red"><u>%2 ft</u></
↪span> )',
  title( lower( "Name" ) ),
  round($length)
)
```

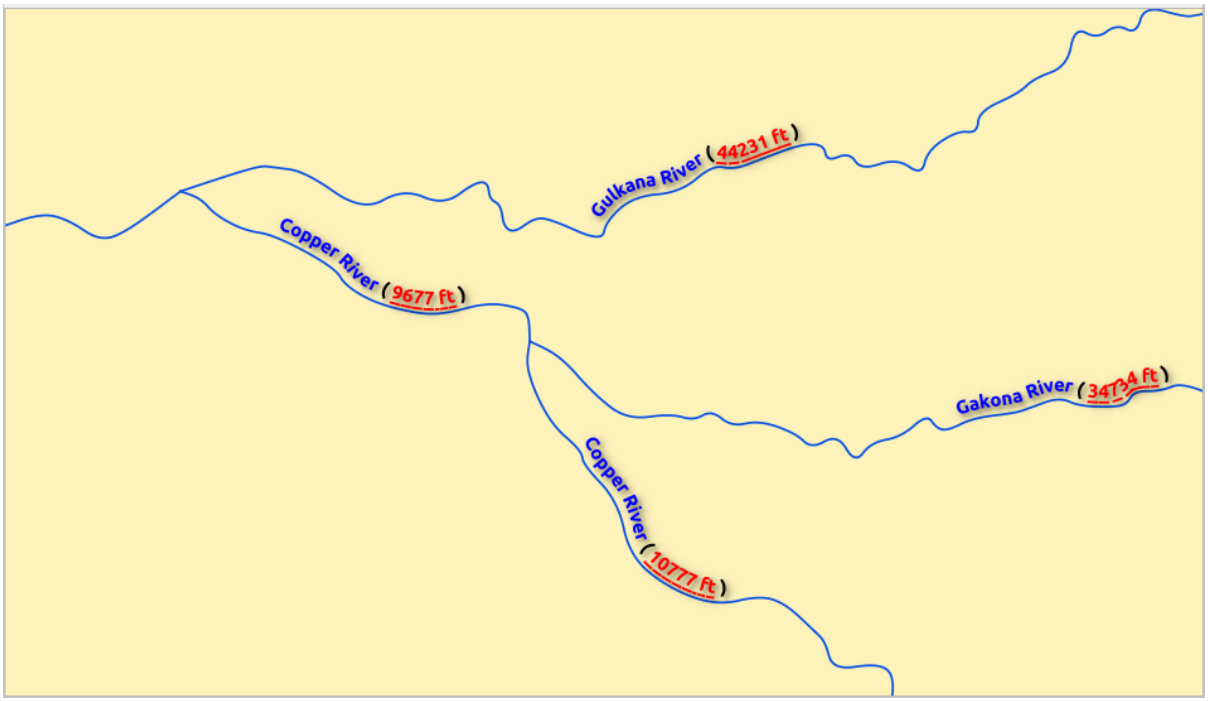


図 14.24: HTML フォーマットを有効にしたラベル

タブの下部にあるウィジェットには **スタイルマネージャデータベース** 内に保存されているアイテムのリストが表示され、このリストはフィルタリング可能です。 これにより、既存のものをベースに現在のテキスト形式やラベルの設定を簡単に設定したり、新しいアイテムをスタイルデータベースに保存したりできます。保存には、 **形式を保存...** または **設定を保存...** ボタンを押して、名前とタグを指定します。

注釈: **ラベルの設定** アイテムを設定する場合、このウィジェットではテキスト形式アイテムも設定可能です。これを選択すると、現在のラベルの **テキストのプロパティ** を素早く上書きすることができます。同様に、そこからテキスト形式を作成/上書きすることもできます。

フォーマット

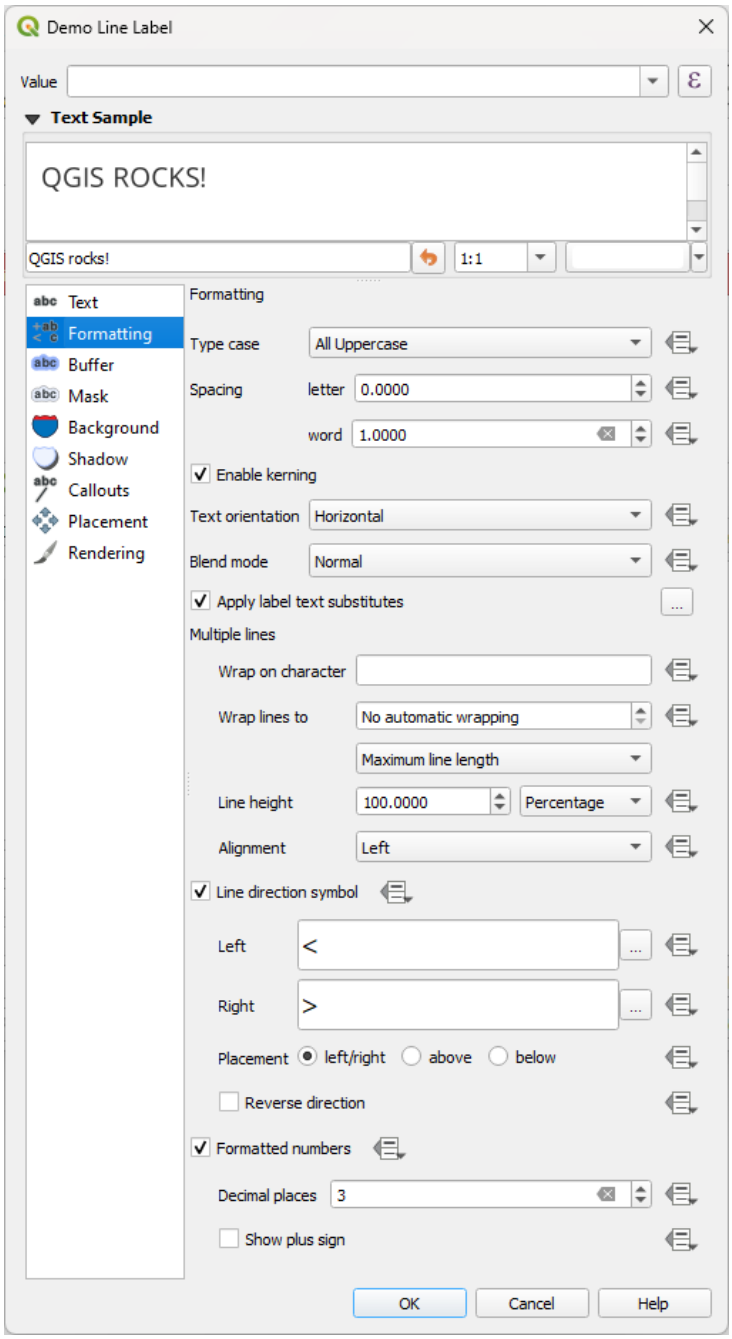


図 14.25: ラベルの設定 - フォーマットタブ

フォーマット タブでは、次の設定を行うことができます:

- **タイプケース** オプションは、テキストの大文字スタイルを変更します。テキストのレンダリング方法は以下のとおりです：
  - **変更なし**
  - **すべて大文字**
  - **すべて小文字**

  - **タイトルのスタイル**：各単語の最初の文字を大文字に修正し、元のテキストが単一のタイプケースを使用している場合には、他の文字を小文字にします。テキストに大文字と小文字が混在している場合には、他の文字はそのままにします。
  - **最初の文字を大文字**：各単語の最初の文字を大文字に修正し、テキスト内の他の文字はそのままにします。

- **間隔** は、単語間および個々の文字間のスペースを変更します。
- *Stretch* 率: テキストの水平方向の伸縮率を指定します。ラベルに余分なテキストを入れるためにフォントの幅を調整するのに便利です。
- テキストフォントの **カーニングを有効にする**
- **テキストの向き** は **水平** または **垂直** を設定できます。ラベルの設定時には、 **回転（***Rotation-based***）** を設定することもできます（例： **線に平行** に配置するモードでライン地物に適切にラベル付けできます）。
- **混合モード** オプションを使用して、ラベルとその下にある地物の混合方法を決定します（詳細は **混合モード** を参照してください）。
- **置換リストを適用** オプションを使用すると、地物ラベルのテキストの代わりになるテキスト（例：ストリートタイプの省略名）のリストを指定することができます。置換テキストは、マップ上にラベルを表示する際に使用されます。ユーザーは置換テキストリストのエクスポートおよびインポートにより、簡単に再利用や共有ができます。
- **複数行** の設定には以下のものがあります：
  - **この文字でラップ処理** オプションを使用して、テキスト内で強制的に改行する文字を設定します
  - **ラップ処理基準** オプションを使用して、理想的な自動改行サイズを指定します。このサイズは **最長行に合わせる** または **最短行に合わせる** とすることができます。
  - **行の高さ**: **ミリメートル**、 **ポイント**、 **ピクセル**、 **パーセント**、 **インチ** の値を設定することができます。行の高さがパーセントに設定されている場合、そのフォントファミリのデフォルトのテキスト行間隔のパーセントになります。通常、テキストサイズの 1.2 倍から 1.5 倍です。
  - **整列** の書式設定： 利用可能な一般的な値は **左**、 **右**、 **正当化する**、 **中央** です。

    ポイントのラベルのプロパティを設定する場合は、テキストの整列を **ラベルの配置に従う** とすることもできます。この場合、テキストの整列はポイントに対するラベルの最終的な配置に依存します。例えば、ラベルがポイントの左側に配置されている場合はラベルが右揃えになり、ポイントの右側に配置されている場合には左揃えになります。

注釈: **複数行** のフォーマットは、線に沿って湾曲する **ラベル配置** にはまだ対応していません。この配置の場合には、このオプションは無効となります。

- ラインに対するラベルの場合は、ラインの方向を決定するのに役立つ **行方向シンボル** を含めることができます。このシンボルは、 **左寄せ** や **右寄せ** を示すために使用します。特に、 **配置** タブで **線に沿って湾曲** や **線に平行に配置** の配置オプションを使用する場合にはうまく機能します。シンボルの配置を設定するオプションや、 **逆方向** にするオプションもあります。

- 数値型テキストのフォーマットを設定するには、 **整形された数値** オプションを使用します。 **小数点以下桁数** の数値を設定できます。デフォルトでは、小数点以下 3 桁です。正の数にプラス記号を表示したい場合には、 **正の符号を表示** にチェックを入れます。

バッファ

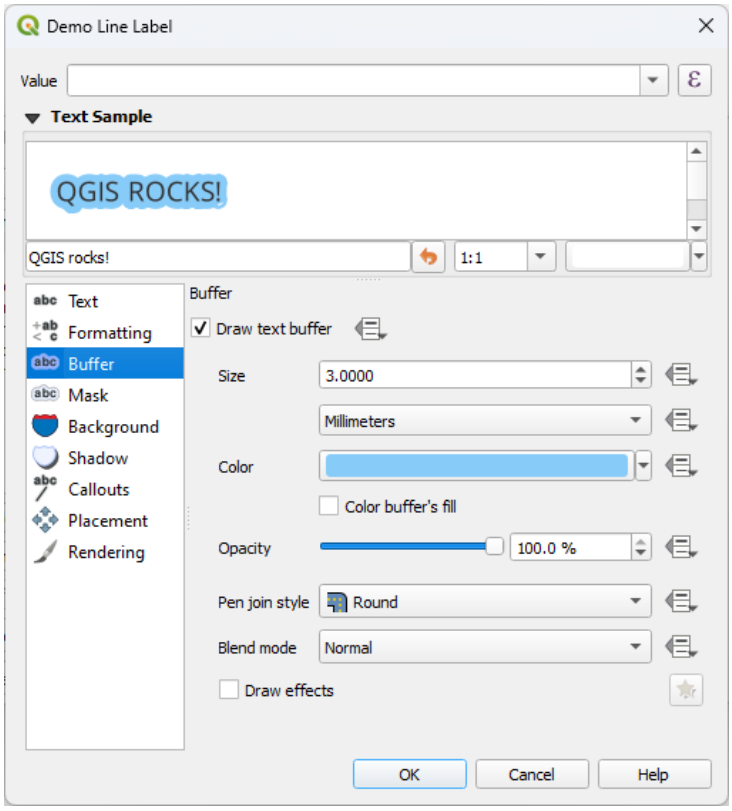


図 14.26: ラベルの設定 - バッファタブ

ラベルの周りにバッファを作成するには、 **バッファ** タブの **テキストバッファを描画** のチェックボックスをオンにします。その後、次の設定ができます。

- バッファの **サイズ** は任意の **サポートされている単位** で設定します。
- バッファの **色** を選択します。
- **カラーバッファの塗りつぶし** ：バッファはラベルのアウトラインから広がるため、このオプションを有効にするとラベルの内部が塗りつぶされます。このオプションは、部分的に透明なラベルを使用する場合や、ラベルのテキストの後ろが見えるような「通常」以外の混合モードを使用している場合に関連性があります。 このオプションをオフにすると、（完全に透明なラベルを使用しているときに）輪郭のあるテキストラベルを作成することができます。
- バッファの **不透明度** を定義する
- **継ぎ目スタイル** ： *Round* 、 *Miter* または *Bevel* から適用できます
- **混合モード** オプションを使用して、ラベルのバッファがマップコンポーネントとどのように混合するかを決定します（詳細については **混合モード** を参照）。

- **描画エフェクト** にチェックを入れると、テキストの読みやすさを向上させるための高度な **描画エフェクト** を追加します。例：アウターグローとぼかし

背景

背景 タブでは、各ラベルの下に表示される図形を設定することができます。背景を追加するには、**背景を描画** のチェックボックスをオンにし、 **図形** タイプを選択します。次の設定ができます：

- **塗りつぶしシンボル** の完全なプロパティを使用した、 **四角形** 、 *Square* 、 **円** 、 **楕円** といった規則的な図形
- ファイル、URL、またはプロジェクトやスタイルデータベースに埋め込み（ **詳細はこちら** ）の *SVG* シンボル
- あるいは、 **シンボルライブラリ** から作成または選択することができる **マーカーシンボル** 。

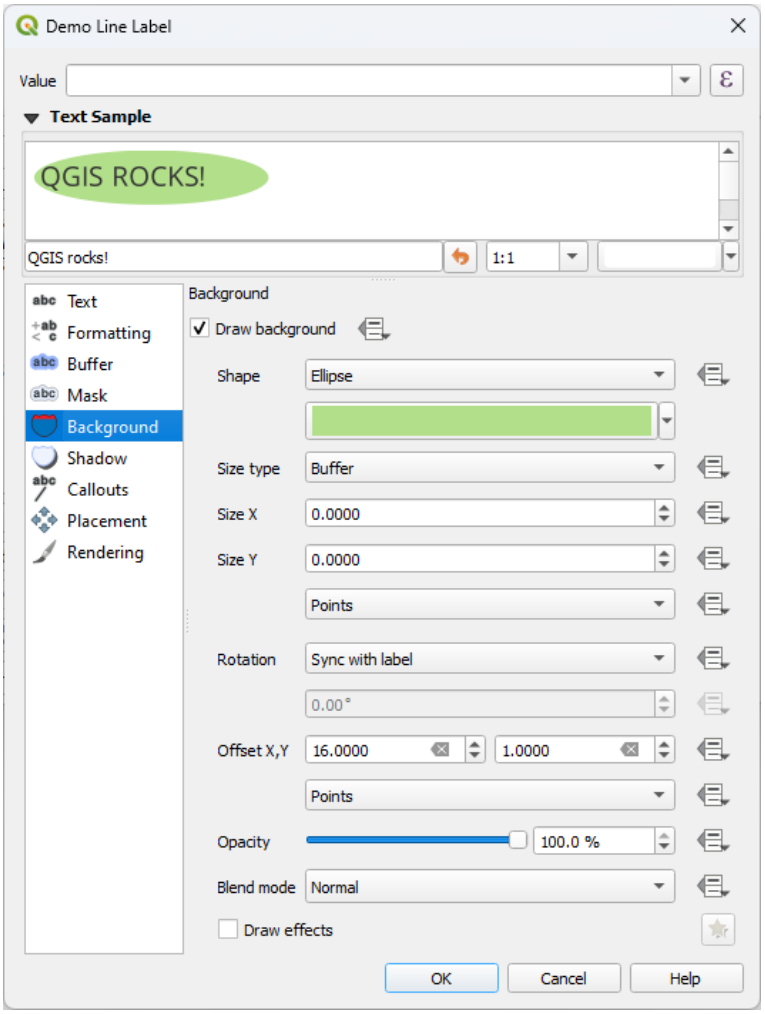


図 14.27: ラベルの設定 - 背景タブ

選択した図形に応じて、以下のプロパティのいくつかを設定する必要があります：

- フレームの **サイズタイプ** には以下の選択肢があります：

  - **一定** ： テキストのサイズとは無関係に、全てのラベルについて同じサイズを使用します
  - **バッファ** ：テキストのバウンディングボックス上のバッファ
- 任意の **サポートされている単位** で表したフレームの X・Y 方向の **サイズ**
- 背景の **回転** ： **ラベルと同期** 、 **ラベルのオフセット** 、 **一定** のどれかです。後の 2 つは度単位の角度の設定が必要です。
- **オフセット量（*X*, *Y*）** ：背景アイテムを X 方向や Y 方向に移動させます
- **半径** *X*, *Y* ：背景図形の角を丸めるための値です（四角形または正方形の図形にのみ適用されます）
- 背景の **不透明度**
- **混合モード** ：レンダリング時の背景と他のアイテムの混合方法の設定です（ **混合モード** を参照してください）。
- SVG シンボルでは、シンボルのデフォルトプロパティを使用する（ **シンボルパラメータをロードする** ）か、カスタムの **塗りつぶし色** 、 **ストローク色** および **ストローク幅** を設定できます。
- **描画エフェクト** ：テキストの読みやすさを向上させるための高度な **描画エフェクト** を追加します。例：アウターグローとぼかし

影

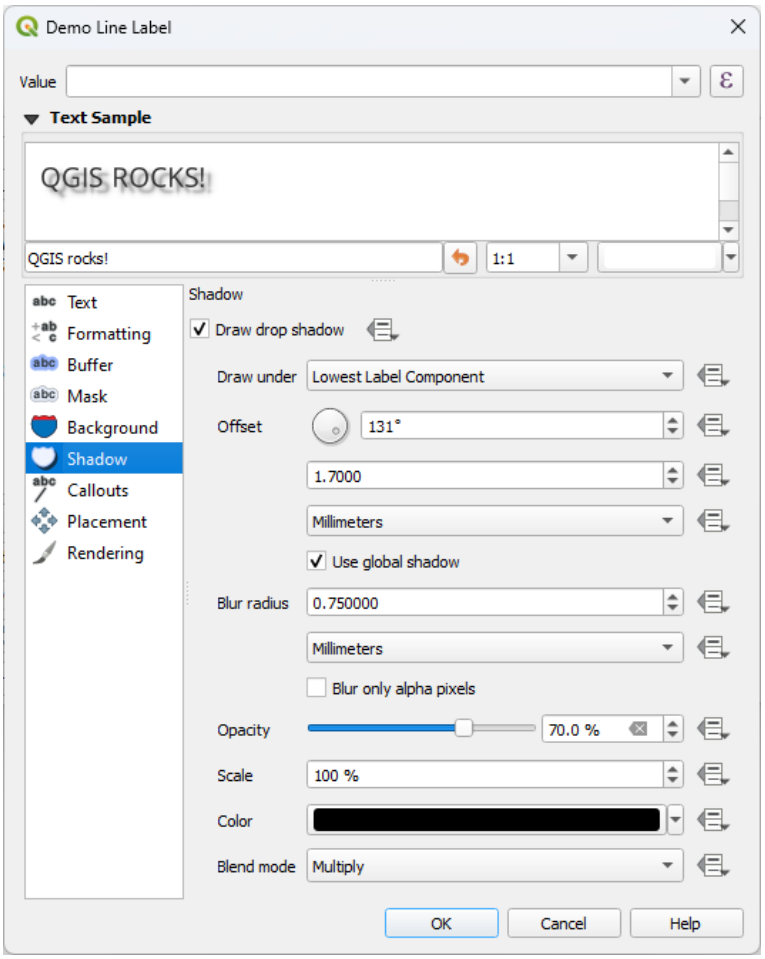


図 14.28: ラベルの設定 - 影タブ

テキストに影を追加するには、 **影** タブを有効にして **影を描画** のチェックボックスをオンにします。その後、次の設定ができます。

- **影の対象** で、影を生成するために使用するアイテムを指定します。これには、**最下位ラベルコンポーネント** 、あるいは **テキスト** 自体、 **バッファ** 、 **背景** 等の特定のコンポーネントを指定できます。
- 影の対象となるアイテムからの **オフセット** を設定します。これには以下の項目があります：
  - 角度： 時計回りの角度を指定します。角度は元のアイテムの方向に依存します。
  - 影の対象となるアイテムからのオフセット距離
  - オフセットの単位

  グローバルシャドウを使用する のチェックボックスにチェックを入れると、角度のゼロ点は常に北向きとなり、ラベルアイテムの向きに依存しなくなります。
- **ぼかし半径** で影の見た目に影響を与えます。数値が大きいほど、影は柔らかくなります。数値の単位は選択することができます。
- 影の **不透明度** を設定します

- **縮尺** 係数を使用して、影のサイズを再スケーリングします
- 影の **色** を選択します
- **混合モード** オプションを使用して、ラベルの影がマップコンポーネントとどのように混合するかを決定します（詳細については **混合モード** を参照）。

14.3.2 ラベルとの相互作用の設定

上述のテキスト書式設定以外にも、ラベルや地物との相互作用を設定することができます。

マスクグリッド（Mask）

マスクグリッド（*Mask*） タブでは、ラベル周囲のマスク領域を定義することができます。この機能は似た色の記号やラベルが重なっている場合に、ラベルを見えるようにしたいときにとても便利です。

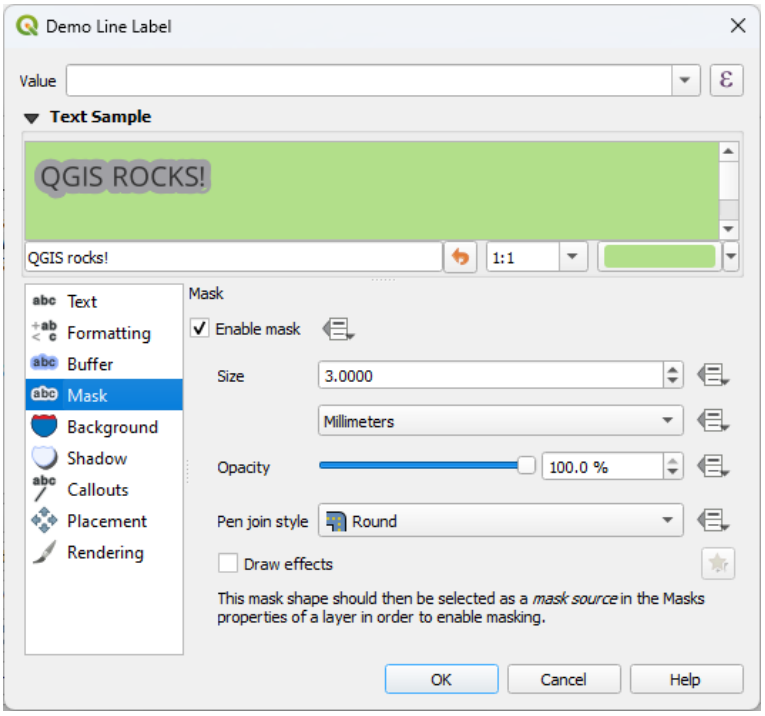


図 14.29: ラベルの設定 - マスクグリッドタブ (テキストサンプルでは、別のレイヤが除外されていることを表す緑色の背景が表示されています)

ラベルにマスク効果を作成するには、

1. タブにある **マスクを有効化** チェックボックスを有効にします。
2. そして、以下の設定を行います
   - **サポートされている単位** でマスクの **大きさ** の設定
   - ラベル周囲のマスク領域の **不透明度**

- **継ぎ目スタイル**
- **描画エフェクト** チェックボックスで設定する **描画効果**

3. 重なってしまうレイヤのプロパティの **マスク** タブ（**マスクプロパティ** 参照）で、このマスク形状をマスクソースとして選択します。

引出し線付きラベル

ごちゃごちゃしたマップにラベルを配置する際の一般的な方法は、**引出し線** を使用することです。関係する地物の外側に配置された（または遠くに移動した）ラベルは、ラベルと地物を結ぶ動的に変化する線によって識別されます。2 つの終端（ラベルまたは地物）のいずれかが移動すると、コネクタの形状は再計算されます。

図 14.30: さまざまな引出し線設定によるラベル

ラベルに引出し線を追加するには、**引出し線付きラベル** タブを有効にして **引出し線付きラベルを描画** のチェックボックスをオンにします。その後、次の設定ができます。

1. コネクタの **スタイル** を以下の中から選びます：
   - **直線** ：最短経路の直線
   - **マンハッタン線** ：90 ° 曲がった折れ線
   - **曲線** ：曲がった線
   - **バルーン** ：ラベルを囲み、地物を指し示す吹出し。角を丸めることもできます。
2. 線ベースの引出し線の場合には：
   1. **線のスタイル** を選択します。描画エフェクトやデータ定義の上書き設定をも含む、**ラインシンボル** の全機能を使用できます。
   2. 曲線の場合には、以下の定義もできます：

- つなぐ線の **曲率** のパーセンテージ
- および **方向** ：ラベルから地物への方向に対して、 **時計回り** 、 **反時計回り** または **自動** （各ラベルについて適切な方向を決定する）とすることができます

3. 引出し線の **最短長** を設定します
4. 地物のラベルから地物の  **全パートに線を描画する** かどうかを指定します
5. **ラベルのアンカー位置** ：コネクタ線がラベルテキストと結合する点を制御します。選択肢は以下のとおりです。
   - **最近傍点**（*Closest Point*）
   - **重心点**
   - 固定された端の位置（ **左上** 、 **中上** 、 **右上** 、 **中央左** 、 **中央右** 、 **左下** 、 **中下** 、 **右下** ）。
6. **ラベルからのオフセット** オプションを設定します：これは、ラベルのアンカーポイント（引出し線の開始位置）からの距離を制御します。これにより、テキストの真上に線を引かないようにできます。

3. バルーンの場合には、以下を設定する必要があります：
   - **塗りつぶしスタイル** 。描画エフェクトやデータ定義の設定も含め、 **塗りつぶしシンボル** の全機能が使えます。
   - 吹出しの **コーナー半径**
   - **ウェッジ幅** ：地物のポインタと吹出しをつなぐ部分の大きさがどれくらいか
   - ラベルのテキスト周囲の **マージン**
4. **地物からのオフセット** オプションを設定します：これは、引出し線の終端位置となる、地物（ポリゴンの場合はそのアンカーポイント）からの距離を制御します。例を挙げると、これにより地物の端の真上に線を引かないようにできます。
5. （ポリゴン）地物の **地物アンカーポイント** （コネクタ線の終点）。選択肢は以下のとおりです。
   - **到達不能極（境界から最も遠い点）**
   - **外殻リング上**（*Point on Exterior*）
   - **内部保証点**（*point on surface*）
   - **重心点**
6. **混合モード** を設定します：引出し線の **混色** を制御します。

データで定義された配置 グループでは、引出し線の **原点** （ラベル側の位置）や **変換先** （地物側の位置）の点座標を制御することができます。また、引出し線は **ラベルツールバー** の  ラベル、ダイアグラム、引出し線を移動 ツールを使用して手動で制御することもできます。各引出し線の始点と終点は、この方法で移動させることができます。マウスポインタが引出し線の始点・終点付近にある場合、強調表示されます。必要に応じて、 `Shift` キーを押しながら移動させることもできます。これにより、引出し線の 2 つの点の間の角度を 15 度単位でスナップさせられます。

配置

ラベルの配置やラベル付けの優先度を設定するには、 **配置** タブを選択します。配置オプションはベクタレイヤの種類、すなわちポイント、ライン、ポリゴンによって異なり、グローバルな **自動配置設定** の影響を受けることに注意してください。

ポイントレイヤのラベルの配置

ポイントラベルの配置モードには、以下のものがあります

- **カルトグラフィックに配置** ：ポイントラベルは以下の理想的なカルトグラフィック配置ルールに従い、ポイント地物とより良い視覚的関係性で生成されます。ラベルは、
  - **サポートされている単位** で設定された **距離** に配置することができます。距離の基準は、ポイント地物自体から、もしくは地物を表すシンボルからの距離です（ **オフセット距離の基準** で設定)。後者のオプションはシンボルサイズが固定されていない場合、例えばサイズがデータによって定義されている場合や、 **カテゴリ値による定義** レンダラで様々なシンボルを使用している場合に特に有用です。
  - **配置の優先度** に従って配置することができます。優先度は、データ定義の優先位置リストを使用して、個別の地物に対してカスタマイズまたは設定することができます。これにより、特定の配置だけを使用することもできますので、例えば、海岸線の地物はラベルが陸側に配置されないようにできます。

    デフォルトでは、カルトグラフィックモードの配置は以下の順序で優先されます（ Krygier and Wood (2011) によるガイドライン や他の地図製作学のテキストを尊重しています)：

    1. 右上
    2. 左上
    3. 右下
    4. 左下
    5. 右
    6. 左
    7. わずかに右寄りの上
    8. わずかに左寄りの下
- **ポイントの周り** ：ラベルは、地物を中心とした等しい半径（ **距離** で設定）の円に配置します。配置の優先順位は「右上」から時計回りです。ラベルの位置はデータ定義の **配置する象限** オプションを使用して制限することもできます。
- **点からのオフセット** ：ラベルはポイント地物から **オフセット量**（*X*, *Y*）離れた位置に配置されます。オフセット量はさまざまな単位で設定でき、望むならば地物の上に重ねることもできます。データで定義された **配置する象限** を使用して配置を制限することができ、ラベルの **回転** を設定することもできます。

ラインレイヤのラベルの配置

ラインレイヤのラベルモードには、以下のものがあります：

- **線に平行に配置** ：地物を表現する一般化されたラインに平行にラベルを描画します。ラベルの位置はラインで直線的な部分が優先されます。以下の設定を定義できます。
  - **許容される位置** ： **線の上**（*Above line*） 、 **線上** 、 **線の下**（*Below line*） そして **線の方向に応じた位置** （ラベルをラインの左または右に配置します）が設定できます。複数のオプションを同時に設定することも可能です。この場合には、QGIS は最適なラベル位置を探します。
  - ラベルとラインの間の **距離**
- **線に沿って湾曲** ライン地物の曲がりに沿ってラベルを描画します。 **線に平行に配置** モードで利用可能なパラメータに加えて、内側や外側の **湾曲表示の場合の文字間最大角度** を設定することができます。
- **水平** ：ラベルを水平に描画します。

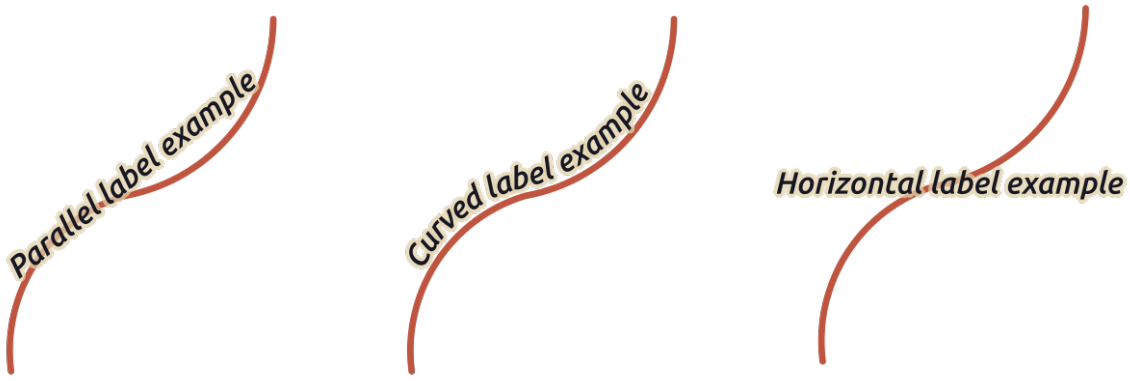


図 14.31: ラインレイヤのラベル配置例

配置モードの設定のほかに、以下の設定ができます：

- **反復** 地物の長さに沿って複数回ラベルを表示するときの **距離** 。この間隔の単位は ミリメートル 、ポイント 、 ピクセル 、 縮尺済みメートル（Meters at Scale） 、 地図単位 、 インチ で入力できます。
- **はみ出し距離 距離** （水平モードでは利用不可）：ラベルがライン地物の終端（または始端）をはみ出す距離の最大許容値を指定します。この値を大きくすると、短いラベル地物でもラベルを表示できるようになります。
- **ラベルのアンカー** ：ラベルが参照するライン地物に沿ったラベルの配置を制御します。 **設定...** をクリックして、以下を選択します：
  - ラベルを近くに配置する、ラインに沿った位置（の比率）。データで定義することもでき、可能な値は以下のとおりです：
    -  **線の中央**
    -  **線の始点**

    * **線の終点**
    * または、 **カスタム**...

- **切り抜き（*Clipping*）**：ラインへのラベルの配置をどのように計算するかを決定します。デフォルトではラインの可視範囲のみを使用しますが、より一貫した結果を得るために、ライン全体を使用することもできます。
- **アンカーポイント**: テキストのどの部分 (開始、中央、終了) をアンカーポイントに合わせるかを制御します。**自動** アンカーを使用すると、以下のようになります：
    * ラインの始点付近（0-25%）にアンカーしたラベルの場合、アンカー位置はラベルテキストの **冒頭** となります
    * ラインの終点付近 (75-100%) にアンカーしたラベルの場合、アンカー位置はラベルテキストの **末尾** となります
    * ラインの中央付近 (25-75%) にアンカーしたラベルの場合、アンカー位置はラベルテキストの **中央** になります
- **配置方針** ： **配置の際に優先するヒント** を使用すると、ラベルのアンカーは単にラベル配置のヒントとして扱われます。 **厳格** を選択すると、ラベルは厳密にラベルアンカーの位置に配置されます。

ポリゴンレイヤのラベルの配置

ポリゴンレイヤのラベル配置は、以下のモードの中から一つ選ぶことができます：

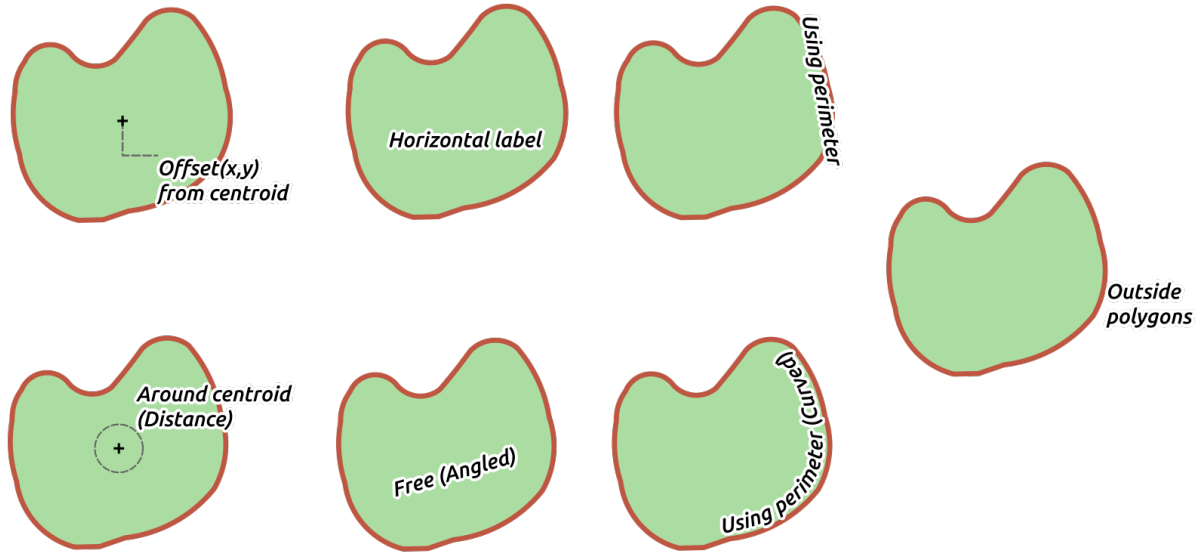


図 14.32: ポリゴンレイヤのラベル配置例

- **重心からのオフセット** ：ラベルは地物の重心上、または重心から固定値で **オフセット量**（*X*, *Y*）だけ離れた距離（ **サポートされている単位** での値）に配置されます。参照する重心点は、マップキャンバスに描画されている部分に基づいた重心（ **ポリゴンの可視部分** ）か、または、見えているかどうかに関係なく **ポリゴンの全体** に基づいた重心により決定されます。また、以下の設定もあります：

  - 重心点をポリゴンの内側に強制移動する
  - ラベルを特定の象限に配置する
  - 回転を指定する
  - **ポリゴンの外側にラベルを配置することを許可** にチェックを入れると、ラベルをポリゴン内部に配置できない場合には外側に配置します。データ定義のプロパティを使用すると、ラベルの外側配置の許可、不許可、強制を地物ごとに設定することが可能になります。

- **重心の周り（*Around Centroid*）** ：ラベルは重心の周りのあらかじめ設定された距離内に配置されます。配置位置は重心位置が優先されます。このモードでも、重心は **ポリゴンの可視部分** か **ポリゴンの全体** のどちらとするかを定義したり、重心点をポリゴンの内側に強制移動したりすることができます。
- **水平** ：ポリゴン内部の最適な場所に水平なラベルを配置します。ポリゴンの端からより遠い場所が優先されます。 **ポリゴンの外側にラベルを配置することを許可** することもできます。
- **自由（回転）** ：ポリゴン内部の最適な場所に回転させたラベルを配置します。ラベルの回転はポリゴンの向きに応じて決められ、ポリゴンの端からより遠い場所への配置が優先されます。 **ポリゴンの外側にラベルを配置することを許可** することもできます。
- **周辺を利用（*Using Perimeter*）** ：ポリゴン境界を表現する一般化されたラインに平行にラベルを描画します。ラベルの位置は周辺のラインの中で直線的な部分が優先されます。以下の設定を定義できます。
  - **許容される位置** ： **線の上（*Above line*）** 、 **線上** 、 **線の下（*Below line*）** そして **線の方向に応じた位置** （ラベルをポリゴンの境界の左または右に配置します）が設定できます。複数のオプションを同時に設定することも可能です。この場合には、QGIS は最適なラベル位置を探します。
  - ラベルとポリゴンの境界の間の **距離**
  - **反復** ポリゴン周囲の長さに沿って複数回ラベルを表示するときの **距離** 。
- **周辺を利用する（湾曲）** ：ポリゴン境界の湾曲に沿ってラベルを描画します。 **周辺を利用** モードで利用可能なパラメータに加えて、内側や外側の **湾曲表示の場合の文字間最大角度** を設定することができます。
- **ポリゴンの外側** ：常にポリゴンの外側の **距離** 分離れた位置にラベルを配置します

共通の配置設定

ラベル配置の設定の一部は、すべてのレイヤジオメトリタイプで利用できます:

ジオメトリジェネレータ

ジオメトリジェネレータ のセクションを使用すると、ユーザーはラベルのレンダリングに使用される基礎となるジオメトリを **式** を使用して変更することができます。これは、ジオメトリを動的に変位させたり、別のジオメトリ（タイプ）に変換するのに便利です。

ジオメトリジェネレータを使用するには：

1. **ジオメトリジェネレータ** のオプションをチェックします
2. 依拠するジオメトリを生成する式を入力します
3. 関連がある場合には、式のアウトプットのジオメトリタイプを選択します。ラベルの配置やレンダリングといった、ジオメトリに基づいたラベルの設定は、新しいジオメトリタイプの機能に合うように更新されます。

以下のような使用例があります：

- "label_position" という別のフィールドに保存されたジオメトリを使用する
- シンボロジの設定で **生成されたジオメトリ** をラベル配置にも使用する
- @map_scale 変数を使用して、距離やサイズの計算をズームレベルとは独立とする
- 湾曲配置モードと組み合わせて、ポイント地物の周囲に丸くラベルを作成する:

```
exterior_ring(make_circle($geometry, 20))
```

- ライン地物の始点と終点にラベルを追加する:

```
collect_geometries( start_point($geometry), end_point($geometry) )
```

- 河川をスムージングしたラインに依存させることで、ラベルの配置空間を増やす:

```
smooth( $geometry, iterations:=30, offset:=0.25, min_length:=10 )
```

データによる定義

データによる定義 グループでは、地物ごとにラベルの配置を直接に制御することができます。これは地物の属性や、設定する式に依存しています。

- *X* 、 *Y* 座標
- 設定したカスタム位置におけるテキストの整列方法
  - **水平方向** ： **Left** 、 **Center** 、 **Right** のいずれかの値をとります
  - **垂直方向** ： **Bottom** 、 **Base** 、 **Half** 、 **Cap** 、 **Top** のいずれかの値をとります
- テキストの **回転** 。ラベルの回転はさまざまな単位（例えば degrees 、 弧の分 、 turns など）で定義できます。フィールドに関連付けられた回転の値を保持し、この回転量をラベルに適用したい場合には、ラベルの固定に無関係に **回転量をプロジェクトに保存** エントリにチェックを入れてくださ

い。チェックを入れない場合には、ラベルの固定を開放すると回転量はリセットされ、値は属性テーブルからクリアされます。

注釈: ポリゴン地物に対するデータ定義の回転は現在のところ、 **重心の周り** 配置モードでのみサポートしています。

注釈: ラベルマップツール（例えば **ラベルを回転** や **ラベルを移動** ツール）と **データ定義の** ラベル配置で同時に式を使用することはできません。ラベルマップツールウィジェットは関連する **補助フィールド** をリセットしてしまいます。

優先度

優先度 セクションでは、各ラベルの配置優先度を定義することができます。すなわち、同じ場所に異なるダイアグラムやラベルの候補がある場合、優先度の高い項目が表示され、他の項目は省略されます。

優先度ランクは、より大きな重みづけを持つ **地物との衝突回避** によってラベルが省略されるかどうかを評価するためにも使用されます。

衝突回避

一部の状況（例：密なラベル、地物が重複する場合）では、無関係な地物の上にラベルが配置されてしまうことがあります。

衝突回避は、他の地物のラベルやダイアグラムを配置してしまわないようにするための QGIS の機能です。これは、 **衝突回避** セクションでコントロールすることができます。

1. **地物と衝突回避** オプションを有効にすると、レイヤの地物が任意のラベルやダイアグラムに対して（同じレイヤ内の他の地物のラベルやダイアグラムも含め）障害物として振る舞うようになります。

   このオプションの横にある データによって定義された上書き コントロールを使用することで、レイヤ全体を対象とするのではなく、一部の地物を選択して障害物として使用することができます。

2. **設定** ボタンを押すと、障害物の重みづけを調整できます。

   - 障害物となる可能性のある地物すべてに対する **衝突回避のウェイト** ：任意の **ラベル** や **ダイアグラム** のうち、配置優先度ランクがこのウェイトよりも大きいものは、上に配置することができます。ランクの低いラベルやダイアグラムは、他に可能な配置がない場合には省略されます。

     この重みづけもデータ定義することができ、同じレイヤ内でも、ある地物は他の地物に比べてより重複しやすくするといったことができます。

   - ポリゴンレイヤでは、地物の障害物の種類を選択できます。

     – **地物の内側** ：ポリゴンの内部にラベルを配置しないようにします（ポリゴンの完全に外側か、わずかに内側に入った位置に配置することを優先します）。

- **地物の輪郭** ：ポリゴンの境界線上にラベルを配置することを避けます（ポリゴンの外側または完全に内側にラベルを配置することを優先します）。これは、地物が全域をカバーしているようなレイヤ（行政単位、カテゴリの範囲など...）の場合に便利です。この場合、これらの地物の内部にラベルを配置するのは不可避であり、地物間の境界にラベルを配置することを避ける方がはるかに見た目が良くなります。

レンダリング

レンダリング タブでは、ラベルがどの状態の時に表示されるかや、他のラベルや地物との相互作用について調整することができます。

ラベルオプション

ラベルオプション では、以下の設定ができます：

- **縮尺に応じた表示設定** や、 **ピクセルに応じた表示設定（地図単位）** があります。
- **ラベルの** *z-index* は、他のレイヤからのラベルと同様に、レイヤ内の他の地物ラベルとの関係性でラベルがレンダリングされる順番を（データ定義の上書き式を使用して）決定します。z-index が高いラベルは、（任意のレイヤの）z-index の低いラベルよりも上にレンダリングされます。

  さらに、2 つのラベルが同じ z-index の場合に対するロジックが以下のように調整されています：

  - 2 つのラベルが同じレイヤのものである場合、小さいラベルが大きいラベルの上に描画されます
  - 2 つのラベルが異なるレイヤのものである場合、ラベルはレイヤ自体の順序と同じように描画されます（つまり、マップ凡例の順序設定を尊重します）

  注釈: この設定は、ラベルを他のレイヤの地物の下に描画するのではありません。すべてのレイヤの地物の上にラベルを描画する順序を制御するだけです。

- **内部配置を許可**： デフォルトでは、QGIS は設定に従って最適な配置でラベルをレンダリングしようとします。 このモードにチェックを入れると、他に選択肢がない場合に、機能がより悪い配置オプションにフォールバックすることを許します。例えば、線が短すぎてカーブしたラベルテキストが入らない場合、ラベルを地物の中心点のすぐ上に水平に配置することができます。
- **表示** および **常に表示するか否かを示す式** にデータによる定義式を使用して、どのラベルをレンダリングするかを細かく調整することができます。
- **逆さま表示** を許可するか：選択肢は **禁止** 、 **回転を定義した場合のみ** 、 **常に** です。
- **重なったラベル** グループでは、レイヤにある地物のラベルに重なりを許すかどうか、またそれぞれのラベルをどのように扱うかを制御できます：
  - **重なりを禁止する**: いくつかのラベルが無くなっても、決してレイヤに重なったラベルを配置しません
  - **必要ならば重なりを許可する**: ラベルを配置できない場合、重なったラベルを描画します。このモードでは、可能であればラベルをあまり理想的でない配置に移動します。例えば、ラベルを

ラインやポリゴンの中心から遠くに移動させるなど、そうすることでラベルの重なりを避けることができる場合です。しかし、他に可能な位置がなければ、ラベルは重なって描画されます。

- **重なりを許容する**: ラベルが他のラベルや障害物に重なっても全く構わずに行われ、重なりを完全に避ける別の配置が可能であっても、常に最良の配置（例えば最も中央の配置）が使われます。

重なったラベルとフォールバック配置オプションの両方を許可することで、レイヤ内のすべての地物がラベル付けされることが保証されます... 必ずしも最高のレンダリングとは限りませんが！

地物オプション

地物オプション では、以下の設定ができます：

- **マルチパートの各パートに表示** するかどうかや、 **ラベル付けする地物の数を制限** するかを選択できます。
- ラインレイヤとポリゴンレイヤには、 **これより地物が小さい場合は省略** を使用して、ラベル付けをする地物の最小サイズを設定するオプションがあります。
- ポリゴン地物については、ラベルが地物の中に完全に収まるか否かによって表示するラベルをフィルタリングすることもできます。
- ライン地物については、 **重複ラベルを除去するために接続する線を結合** を選択でき、 **配置** タブの **距離** や **反復表示の間隔** オプションと組み合わせると、非常にすっきりしたマップをレンダリングできます。

14.4 3D シンボルの作成

スタイルマネージャ では、 各ジオメトリタイプに対して、 *3D* **マップビュー** でレンダリングするための 3D シンボルの作成と保存ができます。

他のスタイル設定と同様に、  *3D* **シンボル** タブをクリックし、  ボタンメニューを展開して、以下のそれぞれについて作成します：

- *3D* **ポイントシンボル**
- *3D* **ラインシンボル**
- *3D* **ポリゴンシンボル**

14.4.1 ポイントレイヤ

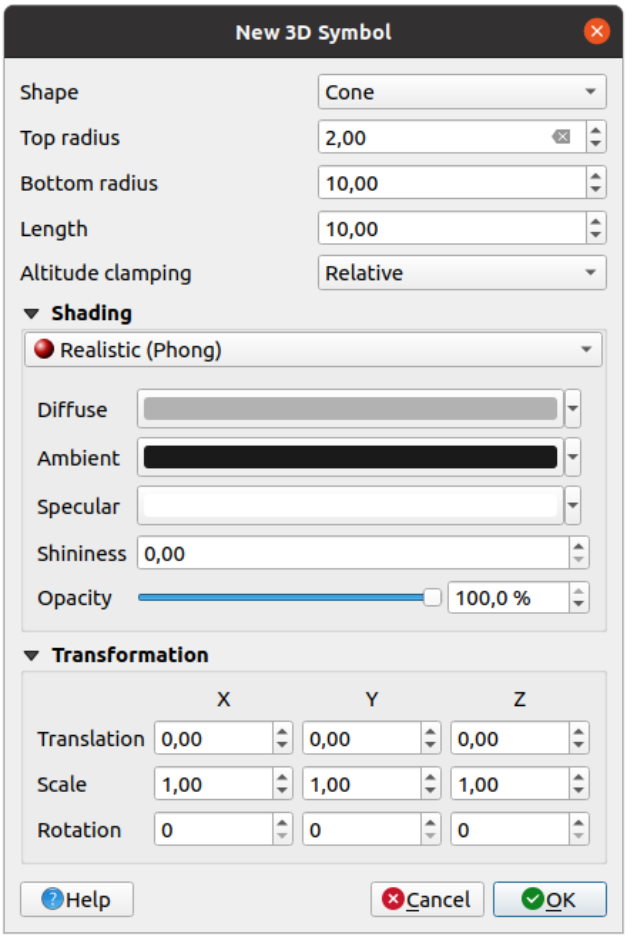


図 14.33: 3D ポイントシンボルのプロパティ

- ポイントシンボルに使用するさまざまな種類の 3D *Shape* を定義できます。これらの Shape では主に寸法を定義し、その単位はプロジェクトの CRS を参照します。利用可能な種類は以下のとおりです：
  - **球** ： **半径** によって定義
  - **円柱** ： **半径** と **長さ** によって定義
  - **立方体** ： **大きさ** によって定義
  - **円錐** ： **上部半径** 、 **下部半径** および **長さ** によって定義
  - **斜面の出力** ： **大きさ** によって定義
  - **トーラス** ： **半径** と **短半径** によって定義
  - *3D* **モデル**、使用する 3D モデルファイル: サポートされているフォーマットは、wavefront `.obj`、`.glTF`、`.fbx` です。モデルはディスク上のファイル、リモートの URL、**プロジェクトの埋め込み** のいずれでも構いません。コミュニティが作成したモデルは QGIS Hub の https://plugins.qgis.org/wavefronts で共有されています。

  - **ビルボード** ： **ビルボードの高さ** と **ビルボードのシンボル** （通常は **マーカーシンボル** に基づく）によって定義。シンボルは一定のサイズとなる。3D 点群形状の可視化に有用

- **標高の制約** は、 **標高の絶対値（*Absolute*）** 、 **地表標高との相対値（*Relative*）** または **地表標高を使う（*Terrain*）** を設定できます。 **標高の絶対値** 設定は、3 次元ベクタレイヤの高さの値が 0 から測った絶対値で与えられている場合に使用します。 **地表標高との相対値** や **地表標高を使う** は、与えられた高さの値を地形の標高値に加えます。
- **シェーディング** プロパティを定義することができます。
- **変換** フレームでは、シンボルにアフィン変換を適用できます。
  - **平行移動（*Translation*）** ：物体を x、y、z 軸方向に移動
  - **縮尺** ：3D 図形をリサイズ
  - **回転** ：x、y、z 軸まわりの回転

14.4.2 ラインレイヤ

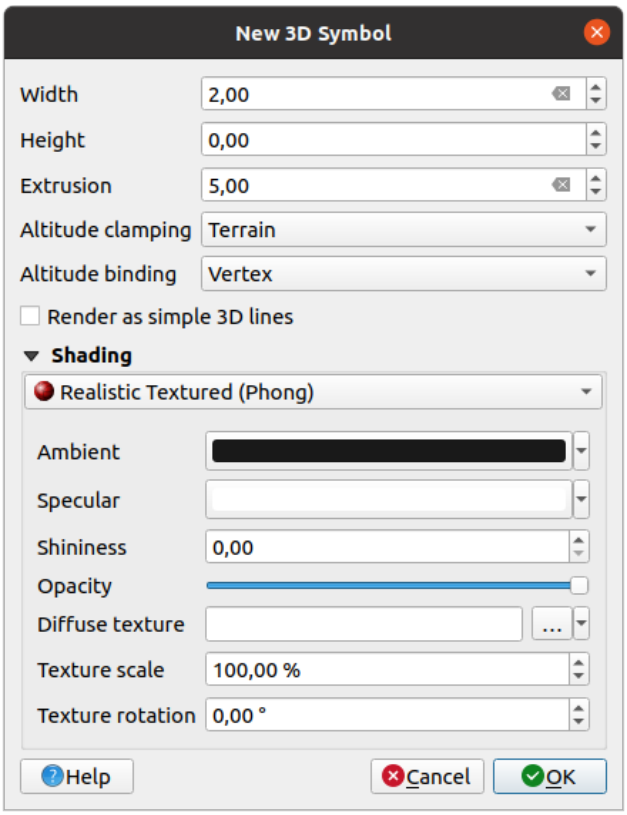


図 14.34: 3D ラインシンボルのプロパティ

- **幅** と **高さ** の設定の下には、ベクタラインの **押出** の設定があります。ラインが z-値を持っていない場合には、この設定で 3D ボリュームを定義することができます。
- 組み込みのラスタ標高データや他の 3D ベクタレイヤがある場合に、 **標高の制約** を使用して地形表面に対する 3D ラインの位置を定義することができます。

- **標高の拘束** は、地物がどのように地形に拘束されるのかを定義します。地物の各 **頂点** で拘束されるか、または **重心** のいずれかです。
- シンプルな *3D* **線として描画** することもできます。
- **シェーディング** プロパティを定義することができます。

14.4.3 ポリゴンレイヤ

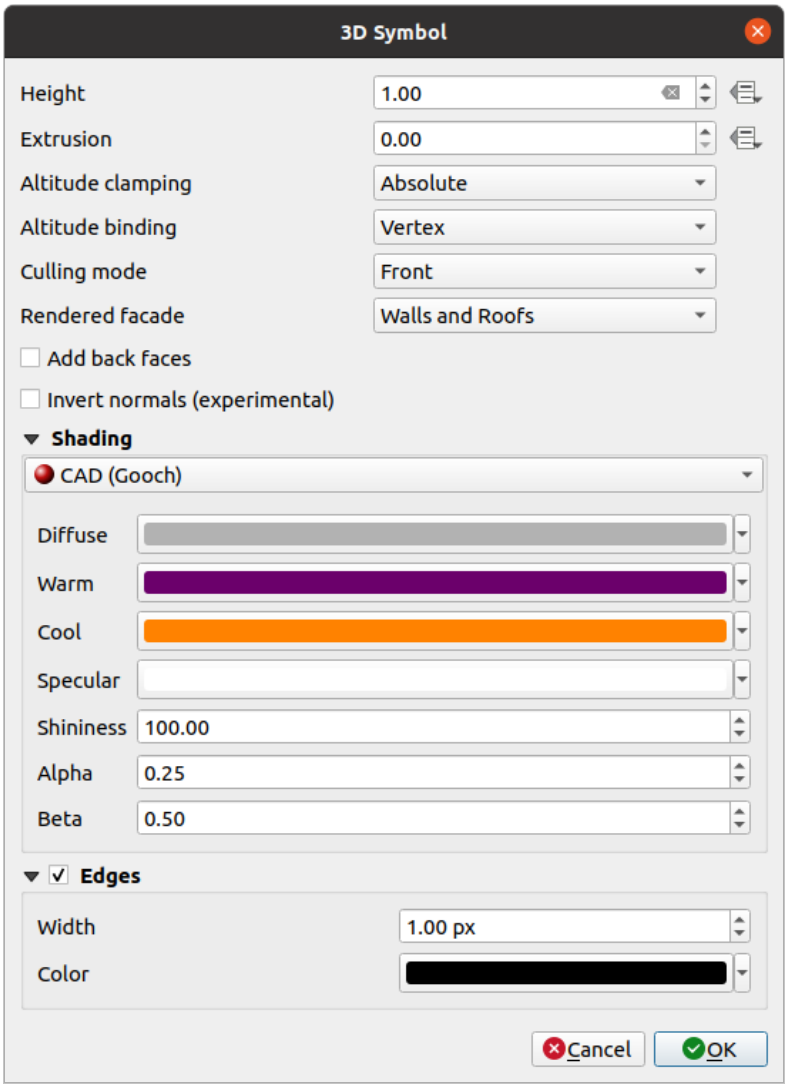


図 14.35: 3D ポリゴンシンボルのプロパティ

- 他の 3D シンボル同様、 **高さ** を CRS の単位で定義できます。また、 ボタンを使用して、カスタム式や変数、属性テーブルのエントリなどで値を上書きすることができます。
- 同様に、z-値の設定が無い場合の **押出** の設定もできます。また、押出にも ボタンが使用でき、ベクタレイヤの値を使って各ポリゴンで異なる押し出し結果を得ることができます。

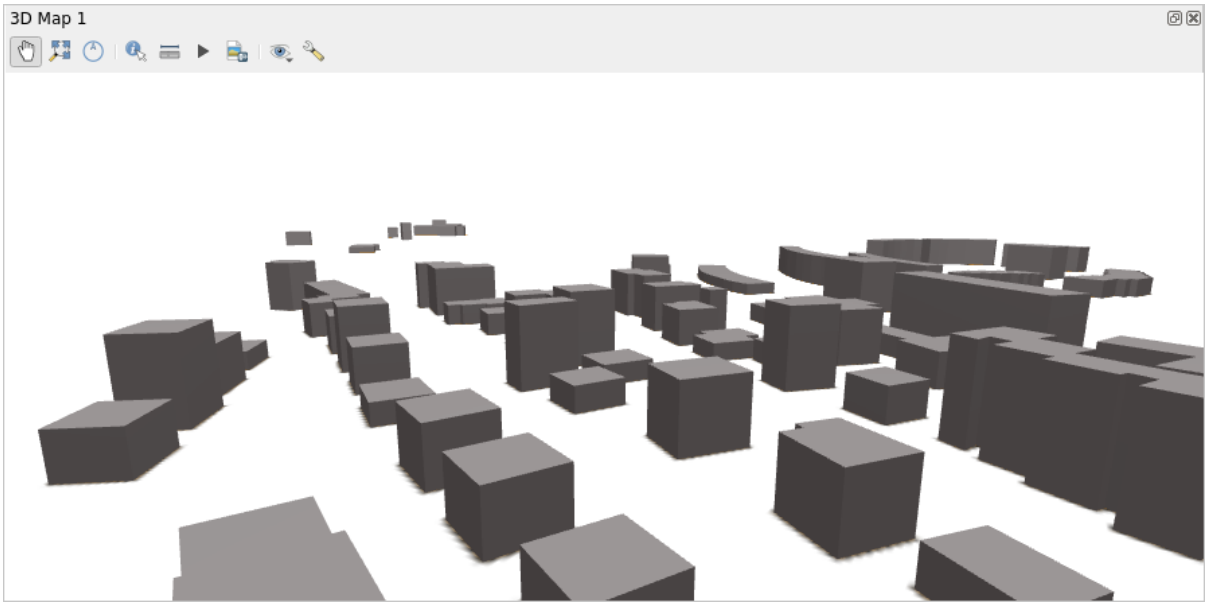


図 14.36: データ定義の押出

- 上で説明したものと同様の **標高の制約** 、 **標高の拘束** を定義することができます。
- シンボルに適用する **カリングモード** には、以下の選択肢があります：
  - *Culling* **なし** ：この設定は、polygonZ/multipatch データの頂点の順序に一貫性（例：全て時計回りあるいは反時計回り）がない場合に、サーフェスが欠けて見えることを防ぎます。
  - **前面カリング**
  - または、 **背面カリング**
- **ファサード** は、表示される面を決定します。可能な値は、 **ファサードなし** 、 **壁** 、 **屋根** 、 **壁と屋根** です。
- **背面を追加** ：各三角形に背面を追加し、表面・裏面の両方に正しい法線を生成しますが、頂点データ数の増加という犠牲を伴います。このオプションは、シェーディングの問題（例えば頂点の順序が整合していないデータに起因する）を修正するために使用できます。
- **法線の反転（試験的）** ：面の頂点の時計回り・反時計回りを修正するのに便利です。
- **シェーディング** プロパティを定義することができます。
- シンボルの **エッジの出力** を有効にすることができ、 **幅** と **色** を指定できます。

ヒント: 3D データのレンダリングのベストな組み合わせ

カリングモード 、 **背面を追加** および **法線の反転** はすべて、3D データの見た目が正しくない場合に修正するためのオプションです。通常は、あるデータを読み込む場合にはまず カリングモード = 反転 と 背面を追加 = 無効 を試してみるのが良いでしょう。これが最も効率的です。レンダリングが正しく表示されない場合には、 背面を追加 = 有効 と、 カリングモード = Culling なし を試してみてください。他の組み合わせはより高度で、入力データセットの頂点順序の混ざり具合に応じた一部のシナリオのみで有用です。

14.4.4 テクスチャのシェーディング

シェーディングは、シーンの照明によって隠れてしまうオブジェクトの 3D ディテールを明らかにするのに役立ちます。地物の可視化にとって適切なシーン照明の設定に悩む必要がないため、結局のところ、作業がより簡単になる設定項目です。

QGIS ではさまざまなシェーディング手法を使用できますが、手法が利用できるかどうかはシンボルのジオメトリタイプに依存します：

- *Phong* **シェーダ**：これは、表面が光を反射する方法を、粗い表面の **拡散光** の反射と、光沢のある表面の **鏡面光** の反射（**シャイネス（輝き）**）の組み合わせとして記述します。また、シーン全体に散乱する弱い光を考慮するための **環境光** のオプションもあります。**不透明度** スライダーを使って、半透明のオブジェクトを 3D でレンダリングします。詳細は https://en.wikipedia.org/wiki/Phong_reflection_model#Description を参照してください。
- *Phong* **シェーダ（テキスチャ）**：*Diffuse* **テキスチャ** として画像が使用される以外は *Phong* **シェーダ** と同じです。画像はディスク上のファイル、リモートの URL、または **埋め込みファイルを抽出** を使用できます。**テキスチャのスケール** と **テキスチャの回転** が必要です。3D で半透明のオブジェクトをレンダリングするには **不透明度** スライダーを使用します。
- *CAD*（*Gooch*）：この手法は、エッジラインやハイライトが視覚的に目立つように、中間色のみでシェーディングを行います。 **拡散光** 、 **鏡面光** 、 **シャイネス** オプションに加えて、 *Warm* カラー（光源に向いている面の色）と *Cool* カラー（光源とは反対にある面の色）を指定する必要があります。また、拡散色による cool と warm の相対的な寄与は、 **アルファ** と *Beta* のプロパティでそれぞれ制御されます。 https://en.wikipedia.org/wiki/Gooch_shading も参照してください。
- **埋め込みテキスチャ**：3D モデルの場合に利用可

14.4.5 適用例

上記で説明した設定を確認するには、https://app.merginmaps.com/projects/saber/luxembourg/tree を見てみましょう。

第15章　データソースの管理

15.1 データを開く

オープンソース・ソフトウェアのエコシステムの一部として、QGIS は様々なライブラリの上に構築されています。これらのライブラリは、QGIS 独自のプロバイダと組み合わされることで、多くのフォーマットを読み込む機能と、そして多くの場合は書き出す機能も提供しています：

- ベクタデータ形式：GeoPackage、GML、GeoJSON、GPX、KML、コンマ区切りテキスト、ESRI フォーマット（シェープファイル、ジオデータベース等）、MapInfo、MicroStation ファイル形式、AutoCAD DWG/DXF、GRASS、その他多くの形式が含まれます。完全なリストは、 サポートしているベクタ形式 を参照してください。
- ラスタデータ形式： GeoTIFF、JPEG、ASCII Gridded XYZ、MBTiles、R や IDRISI ラスタ、 GDAL Virtual、SRTM、Sentinel Data、ERDAS IMAGINE、ArcInfo Binary Grid、ArcInfo ASCII Grid、その他多くの形式が含まれます。完全なリストは サポートしているラスタ形式 を参照してください。
- データベース形式：PostgreSQL/PostGIS, SQLite/SpatiaLite, Oracle, MS SQL Server, SAP HANA, MySQL 等
- Web マップと Web データサービス（WM(T)S、WFS、WCS、CSW、XYZ tiles、ArcGIS services 等）も QGIS のプロバイダで扱うことが可能です。これらの一部についてのより詳しい情報は *OGC / ISO* **プロトコルで作業する** を参照してください。
- アーカイブされたフォルダからサポートしているフォーマットのファイルを読み込んだり、QML ファイル（ *QML-QGIS* **スタイルファイル形式** ）などの QGIS ネイティブ形式や仮想レイヤ、メモリレイヤを使用することができます。

80 以上のベクタ形式と 140 以上のラスタ形式が GDAL と QGIS ネイティブプロバイダによってサポートされています。

注釈: さまざまな理由により、リストに挙げたすべてのフォーマットが QGIS で動作するわけではありません。例えば、外部のプロプライエタリなライブラリを必要とする場合や、お使いの OS の GDAL/OGR のインストールが、使用したいフォーマットをサポートするようにビルドされていない場合もあります。利用可能なフォーマットのリストを確認するには、コマンドラインで `ogrinfo --formats` （ベクタ用）や `gdalinfo --formats` （ラスタ用）を実行するか、QGIS 上で **設定**　**オプション**　*GDAL* のメニューを確認してください。

QGIS では、データ形式に応じて、データセットを開くためのさまざまなツールがあります。それらは主に **レイヤ**　**レイヤの追加**　メニューまたは **レイヤ管理** ツールバー（ **ビュー**　**ツールバー** メニューで有効にできます）で使用できます。ただし、これらのツールはすべて **データソースマネージャ** ダイアログというただ一つのダイアログを指しています。このダイアログは **データソースマネージャツールバー** にあ

る  データソースマネージャを開く ボタンを押すか、または Ctrl+L を押すと開くことができます。 **データソースマネージャ** ダイアログ（ 図 15.1 ）は、ファイルベースのデータやデータベース、QGIS がサポートする Web サービスを開くための統一されたインターフェイスを提供します。

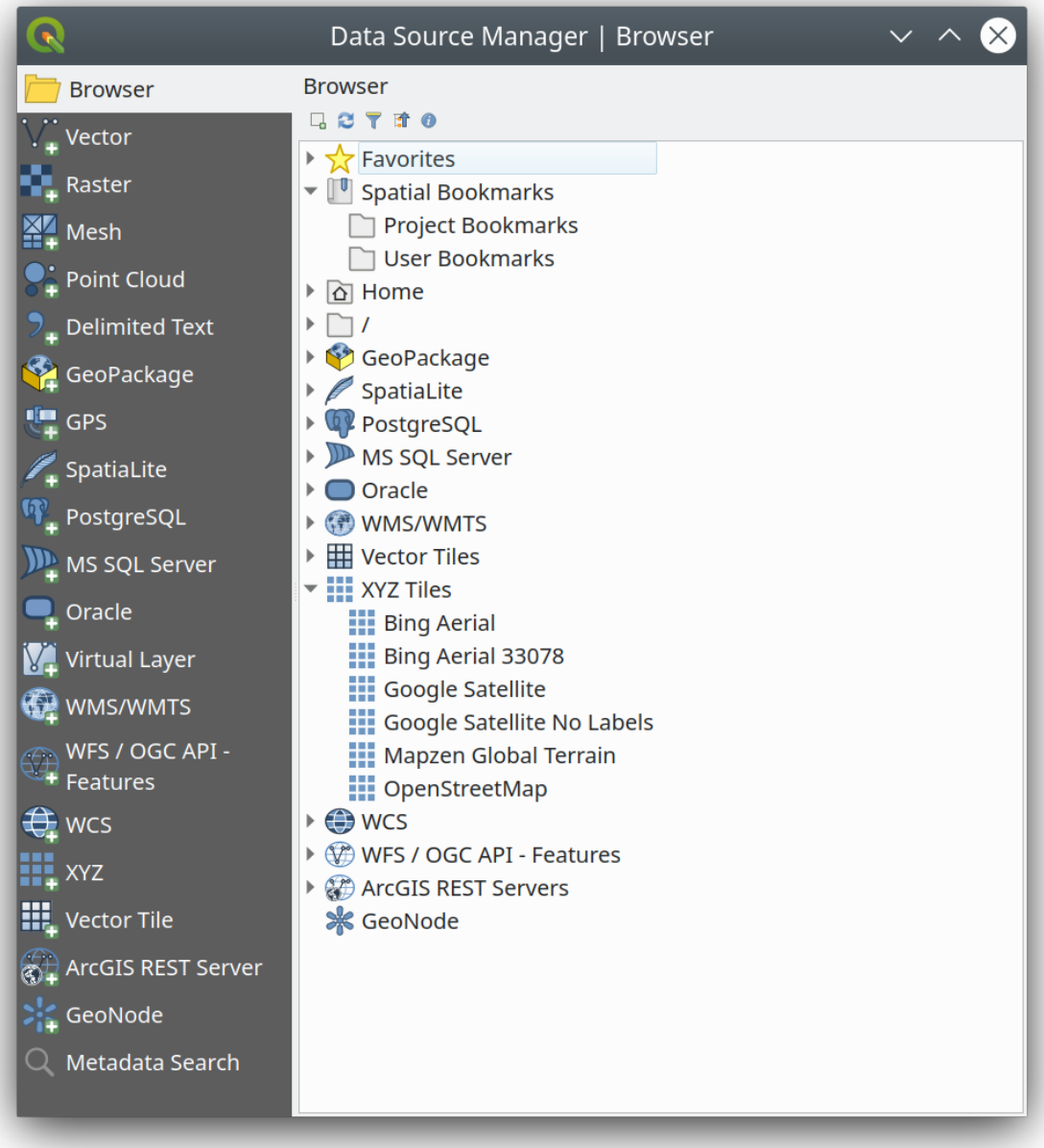


図 15.1: QGIS のデータソースマネージャダイアログ

このメインエントリポイントの他に、接続されたデータベースを分析したり操作したりする高度な機能を提供する *DB* **マネージャ** プラグインがあります。DB マネージャの機能についての詳細は、*DB* **マネージャプラグイン** を参照してください。

他にも多くのツールやネイティブのプラグイン、サードパーティ製のプラグインがあり、様々なデータ形式を開くのに役立ちます。

この章では、QGIS でデータをロードするためにデフォルトで提供されているツールのみを説明します。主

に、 **データソースマネージャ** ダイアログに焦点を当てますが、 各タブの説明だけでなく、データプロバイダやフォーマットの仕様に基づいたツールについても説明します。

15.1.1 ブラウザパネル

ブラウザ は、素早く簡単にデータをプロジェクトに追加する主な方法の一つです。これは、以下の方法で利用できます。

-  データソースマネージャを開く ボタン（ Ctrl+L ）を押して開いた **データソースマネージャ** のタブの一つにブラウザがあります。
- **ビュー　パネル**（  は **設定　パネル**）メニューもしくは Ctrl+2 を押すことで開く QGIS のパネルとして利用できます。

どちらの場合でも、 **ブラウザ** はレイヤの種類（ラスタ、ベクタ、テーブル）やデータソースの形式（プレーンまたは圧縮ファイル、データベース、web サービス）に関係なく、ファイルシステム内を検索しジオデータを管理するのに役立ちます。

インターフェースについて

ブラウザパネルの上部には、以下の機能を持つボタンがあります：

-  選択したレイヤの追加 ：レイヤのコンテキストメニューから **レイヤをプロジェクトに追加する** を選択することでも、マップキャンバスにデータを追加することができます。
-  再読み込み ：ブラウザツリーを更新します
-  ブラウザのフィルタ ：特定のデータを検索するためにブラウザをフィルタリングします。検索語やワイルドカードを入力すると、ブラウザはツリーをフィルタリングして、一致する DB テーブル、ファイル名、フォルダへのパスのみを表示します -- 他のデータやフォルダは表示されません。 図 15.2 のブラウザ (2) パネルの例を参照してください。比較は大文字と小文字を区別するかどうかを指定できます。また、下記のオプションを設定することもできます：
  - **通常** ：検索テキストを含むアイテムを表示します
  - **ワイルドカード** ：? や * の文字を使用して、検索テキストの位置指定を調整した検索を実行できます。
  - **正規表現**
-  すべて折りたたむ ：ツリー全体を折りたたみます
-  プロパティウィジェットの有効化/無効化 ：オンに切り替わると、パネルの下部に新しいウィジェットが追加され、選択されたアイテムのメタデータが表示されます。

ブラウザ パネル内のエントリは階層的にまとめられており、いくつかのトップレベルエントリがあります：

1. **お気に入り** ：よく使用する場所へのショートカットを置くことができる場所です
2. **空間ブックマーク** ：よく使用するマップ範囲を保存できる場所です（ **地図上の範囲のブックマーク** 参照）

3. **プロジェクトホーム** ：プロジェクトに関連したデータ（の大半）が保存されているフォルダへのクイックアクセスです。デフォルト値は、プロジェクトファイルがあるディレクトリです。
4. ファイルシステムの **ホーム** ディレクトリと、ファイルシステムのルートディレクトリ
5. 接続されているローカルドライブやネットワークドライブ
6. それから、プラットフォームや基礎となるライブラリにもよりますが、多数のコンテナ / データベース形式やサービスプロトコルのトップレベルエントリがあります：
   -  *GeoPackage*
   -  *SpatiaLite*
   -  *PostgreSQL*
   -  *SAP HANA*
   -  *MS SQL Server*
   -  *Oracle*
   -  *WMS/WMTS*
   -  *Vector Tiles*
   -  *XYZ Tiles*
   -  *WCS*
   -  *WFS/OGC API-Features*
   -  *ArcGIS REST Server*
   -  *GeoNode*

ブラウザ項目とのインタラクション

ブラウザは、ブラウザへのドラッグ＆ドロップや、ブラウザからキャンバスや **レイヤ** パネルへのドラッグ＆ドロップ、 **レイヤ** パネルからブラウザ内のレイヤコンテナ（例えば GeoPackage 等）へのドラッグ＆ドロップをサポートしています。

ブラウザ内のプロジェクトファイルのアイテムは展開することができ、プロジェクトに含まれる（グループを含む）完全なレイヤツリーを表示します。プロジェクトのアイテムはブラウザ内の他のアイテムと同様に扱われるため、（例えば GeoPackage ファイルにレイヤアイテムをコピーするために）ブラウザ内でドラッグ＆ドロップしたり、ドラッグ＆ドロップやダブルクリックで現在のプロジェクトにアイテムを追加したりすることができます。

ブラウザ パネル内の要素を右クリックすると、コンテキストメニューが開きます。

ファイルシステムのディレクトリエントリの場合、コンテキストメニューには次のようなものがあります：

- **新規** は、下記のエントリから選択して新規作成します

  - **ディレクトリ**...
  - *GeoPackage*...
  - **シェープファイル**...

- **お気に入りとして追加** ：お気に入りフォルダは好きな時に名前の変更（ **お気に入りの名前の変更**... ） や削除（ **お気に入りを削除** ）ができます。
- **ブラウザから隠す** ：隠されたフォルダは **設定** **オプション** **データソース** **ブラウザに表示しないパス** の設定で見えるように戻すことができます。
- **このディレクトリの高速スキャン**
- **ディレクトリを開く**...
- **ターミナルで開く**...
- **プロパティ**...
- **ディレクトリプロパティ**...

プロジェクト内でレイヤとして扱うことのできるリーフエントリについては、コンテキストメニューにサポート項目があります。例えば、非データベース、非サービスベースのベクタ、ラスタ、メッシュデータソースには、以下の項目があります：

- **レイヤエクスポート** **ファイルへ**...
- **レイヤをプロジェクトに追加する**
- **レイヤのプロパティ**
- **管理** *"<name of file>"* **の名前を変更**... または *"<name of file>"* **を削除**...
- **ファイルに表示**
- **ファイルプロパティ**...

レイヤのプロパティ エントリを選択すると、（レイヤがプロジェクトに追加された後に **ベクタ** レイヤや **ラスタ** レイヤのプロパティで表示されるものと同様な）以下の情報が表示されます：

- レイヤに関する **メタデータ** 。メタデータのグループは次のとおりです： **プロバイダからの情報** （可能ならば **パス** はソースへのハイパーリンクで表示されます）、 **識別** 、 **領域** 、 **アクセス** 、 **属性** （ベクタレイヤの場合）、 **バンド** （ラスタレイヤの場合）、 **連絡先** 、 **リンク** （ベクタレイヤの場合）、 **リファレンス** （ラスタレイヤの場合）、 **履歴**
- **プレビュー** パネル
- ベクタソースの場合は属性テーブル（ **属性** パネル内）

ブラウザ を使用してプロジェクトにレイヤを追加するには、次の手順で操作します：

1. **ブラウザ** を上記の方法で有効にします。ファイルシステム、データベース、web サービスのブラウザツリーが表示されます。データベースや web サービスが表示される前に、データベースや web サービスへ接続する必要があるかもしれません（専用のセクションを参照してください）。
2. リストの中からレイヤを見つけます。

3. レイヤのコンテキストメニューを使用するか、レイヤ名をダブルクリックするか、レイヤを **マップキャンバス** へとドラッグ＆ドロップします。これでレイヤは **レイヤパネル** に追加され、マップキャンバス上で見ることができるようになります。

Tip: ブラウザから QGIS プロジェクトを直接開く

プロジェクト名をダブルクリックするか、マップキャンバス上へドラッグ＆ドロップすることで、ブラウザパネルから QGIS のプロジェクトを直接開くこともできます。

ファイルがロードされたら、マップナビゲーションツールを使用してレイヤをあちこちズームすることができます。レイヤのスタイルを変更するには、凡例内のレイヤ名をダブルクリックするか、レイヤ名を右クリックしてコンテキストメニューから **プロパティ** を選択し、 **レイヤプロパティ** ダイアログを開きます。ベクタレイヤのシンボロジ設定の詳細については、 **シンボロジプロパティ** を参照してください。

ブラウザツリー内のアイテムを右クリックすると、以下の機能があります：

- ファイルやテーブルについては、メタデータを表示したり、プロジェクト内に開きます。テーブルは名前を変更したり、行を削除したり、全行削除することもできます。
- フォルダについては、お気に入りにブックマークしたり、ブラウザツリーから隠したりできます。非表示となったフォルダは、 **設定** **オプション** **データソース** タブで管理することができます。
- **空間ブックマーク** の管理：ブックマークを作成したり、ブックマークを `XML` ファイルとしてエクスポートしたり、`XML` ファイルからインポートしたりできます。
- データベースや web サービスへの接続を作成します。
- 再読み込みや、スキーマ名の変更、スキーマの削除ができます。

また、ファイルをデータベースにインポートしたり、1 つのスキーマ/データベースから別のスキーマ/データベースにテーブルをコピーしたりすることもドラッグ＆ドロップで簡単に行うことができます。2 つ目のブラウザパネルを利用することで、ドラッグ中に長いスクロールを行わずに済みます。ファイルを選択して、一方のパネルから他方のパネルへとドラッグ＆ドロップするだけです。

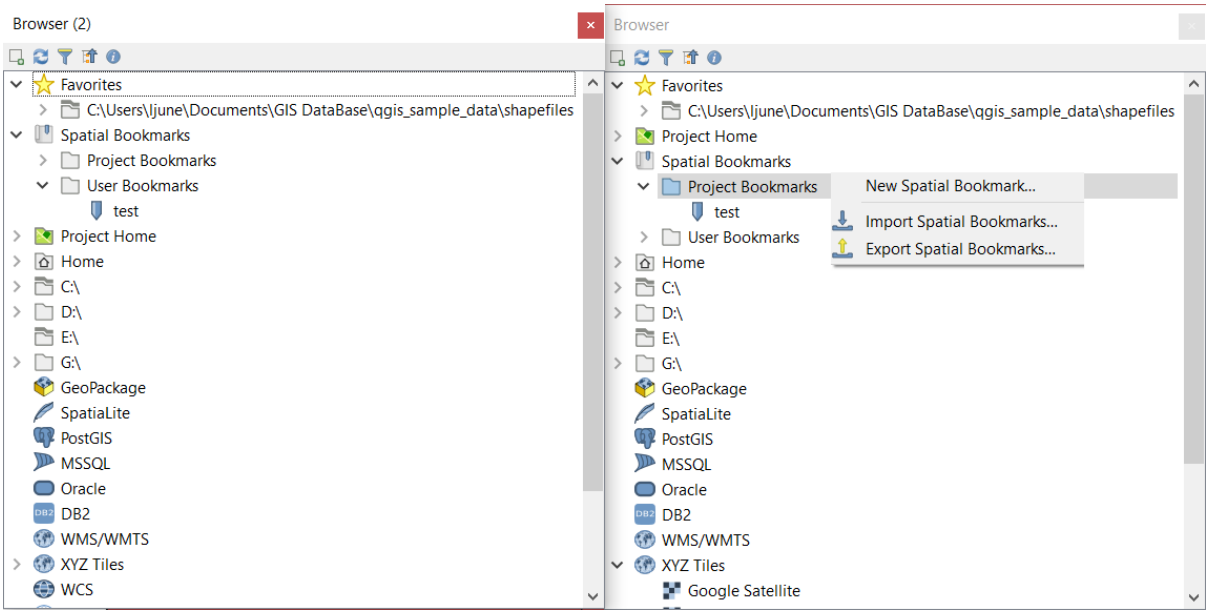


図 15.2: 横並びの QGIS ブラウザパネル

Tip: OS のファイルブラウザからドラッグ＆ドロップで QGIS にレイヤを追加する

オペレーティングシステムのファイルブラウザから **レイヤパネル** またはマップキャンバスにファイルをドラッグ＆ドロップしても、ファイルをプロジェクトに追加できます。

15.1.2 DB マネージャ

DB **マネージャ** プラグインは、QGIS がサポートする空間データベース形式（PostGIS、SpatiaLite、GeoPackage、Oracle Spatial、MS SQL Server、仮想レイヤ）を統合して管理するためのもう一つのツールです。これは、**プラグイン** **プラグインの管理とインストール...** メニューから有効化することができます。

DB マネージャ プラグインにはさまざまな機能があります。

- データベースに接続し、その構造や内容を表示する
- データベースのテーブルをプレビューする
- ダブルクリックまたはドラッグ＆ドロップで、マップキャンバスへレイヤを追加する
- QGIS ブラウザや他のデータベースから、データベースにレイヤを追加する
- SQL クエリを作成し、実行結果をマップキャンバスへ追加する
- **仮想レイヤ** を作成する

DB マネージャの機能に関する更なる情報は、 *DB* **マネージャプラグイン** を参照してください。

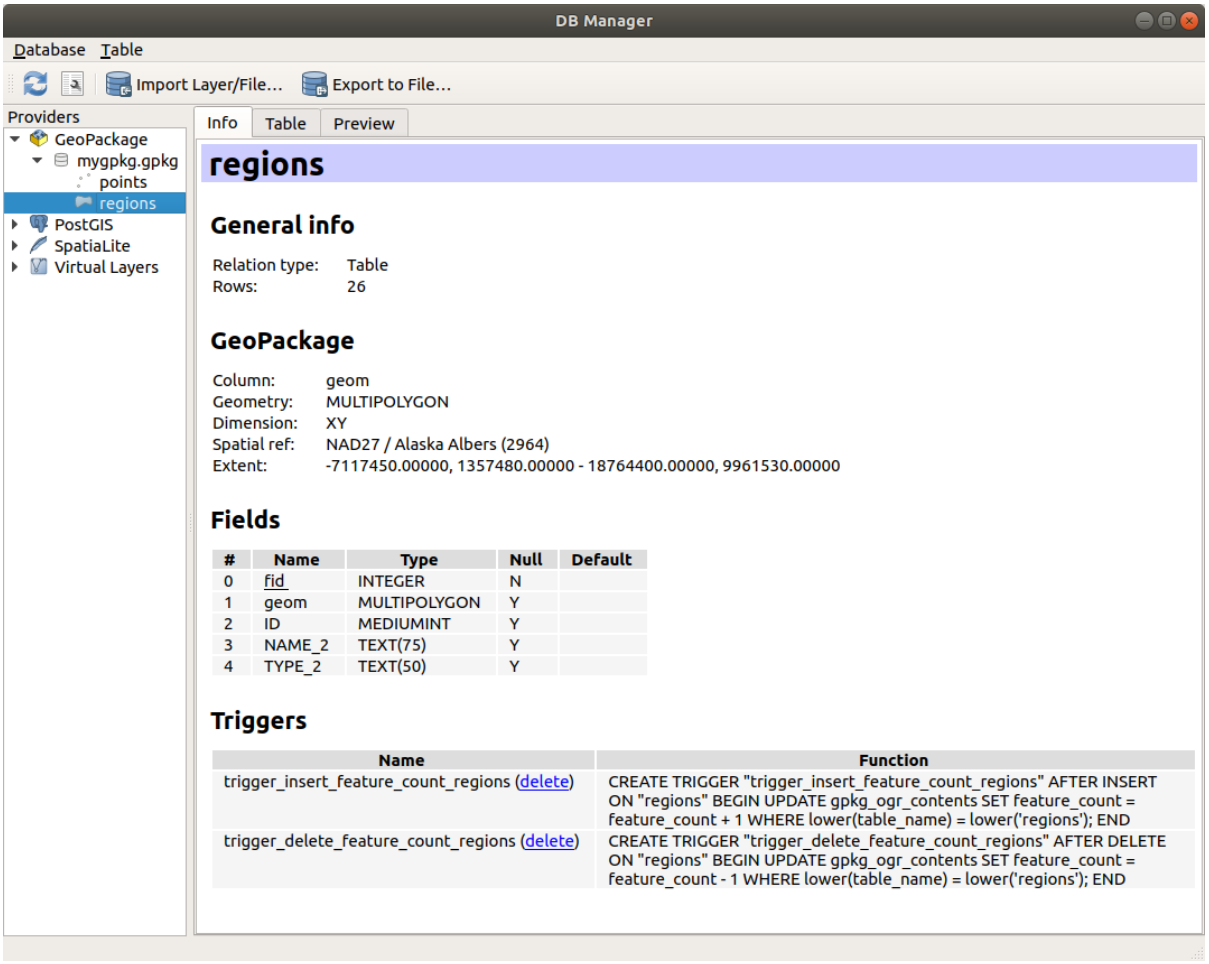


図 15.3: DB マネージャダイアログ

15.1.3 プロバイダベースの読み込みツール

ブラウザパネルと DB マネージャはレイヤを追加するために QGIS が提供する主なツールですが、この他に、データプロバイダに特化したツールもあります。

注釈: いくつかの **外部プラグイン** も、QGIS で特定の形式のファイルを開くためのツールを提供しています。

ファイルからレイヤを読み込む

ファイルからレイヤを読み込むには、次の手順で操作します：

1. **データソースマネージャ** ダイアログでレイヤタイプのタブを開きます。つまり、 
データソースマネージャを開く ボタン（または Ctrl+L ）を押し、目的のタブを有効にします。 もしくは、

   - ベクタデータ（GML、ESRI シェープファイル、Mapinfo、DXF など）については、Ctrl+Shift+V を押すか、 **レイヤ** **レイヤの追加**  **ベクタレイヤの追加** メニューオプションを選択するか、 
ベクタレイヤの追加 ツールバーボタンをクリックします。

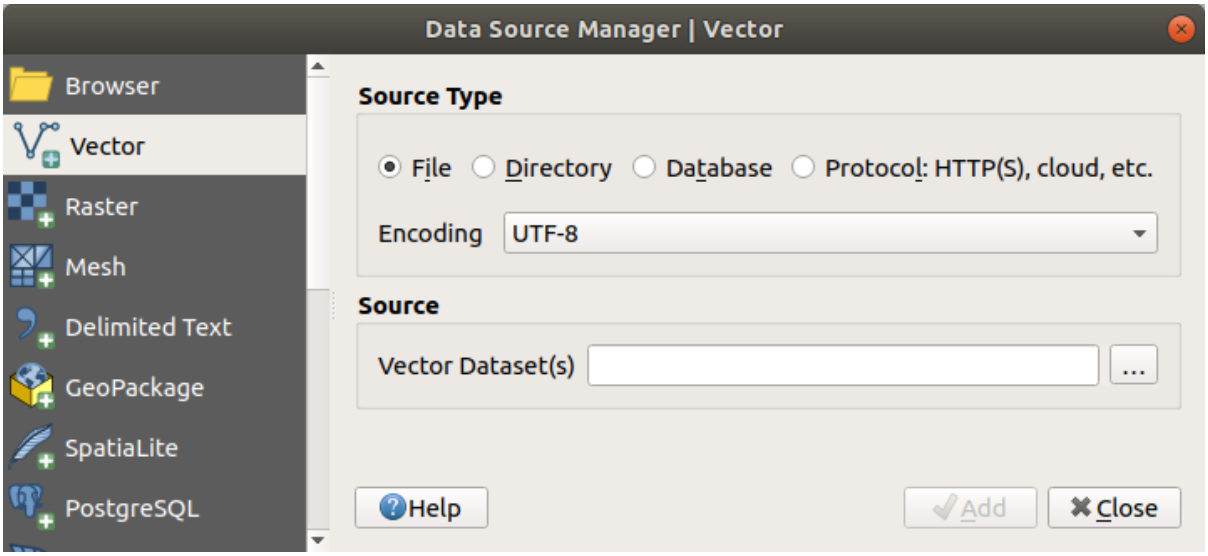


図 15.4: ベクタレイヤの追加ダイアログ

   - ラスタデータ（GeoTiff、MBTiles、GRIdded Binary、DWG など）については、 Ctrl+Shift+R を押すか、 **レイヤ** **レイヤの追加**  **ラスタレイヤの追加** メニューオプションを選択するか、 
ラスタレイヤの追加 ツールバーボタンをクリックします。

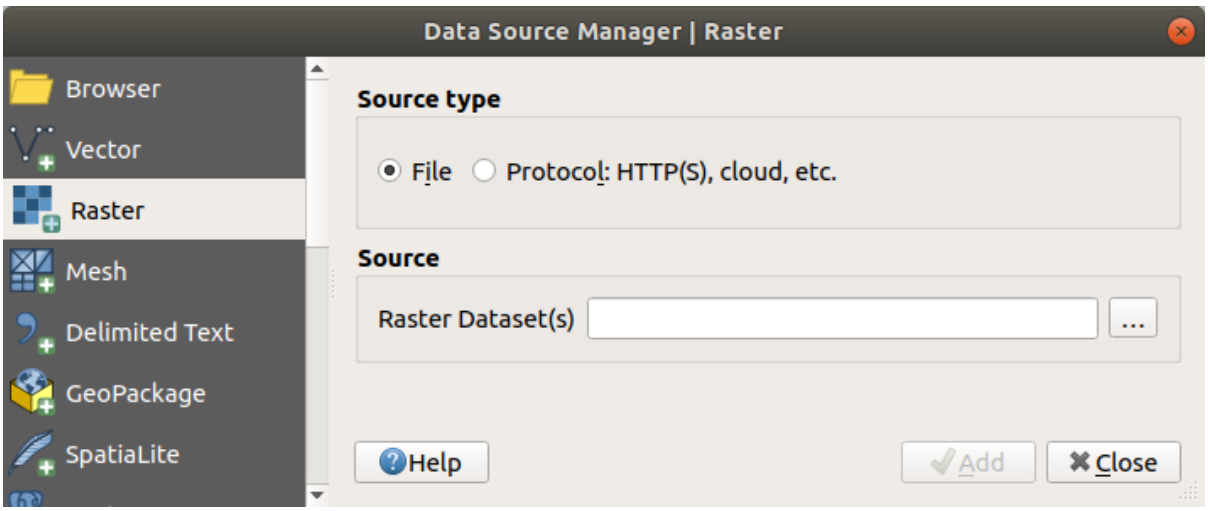


図 15.5: ラスタレイヤの追加ダイアログ

2. ソースタイプは **ファイル** を選択します
3. ... ブラウズ ボタンをクリックします
4. ファイルシステムを検索し、サポートされているデータソースを読み込みます。ダイアログで `Ctrl` キーを押しながら複数のアイテムをクリックするか、 `Shift` キーを押しながら最初と最後のアイテムを選択することでアイテム範囲を選択することで、複数のレイヤを同時に読み込むことが可能です。ファイル形式のフィルタには、十分にテストされたファイル形式のみが表示されます。他のファイル形式は、 全ファイル （プルダウンメニューの一番上のアイテム）を選択することで読み込むことができます。
5. **開く** ボタンを押し、選択したファイルを **データソースマネージャ** ダイアログに読み込みます

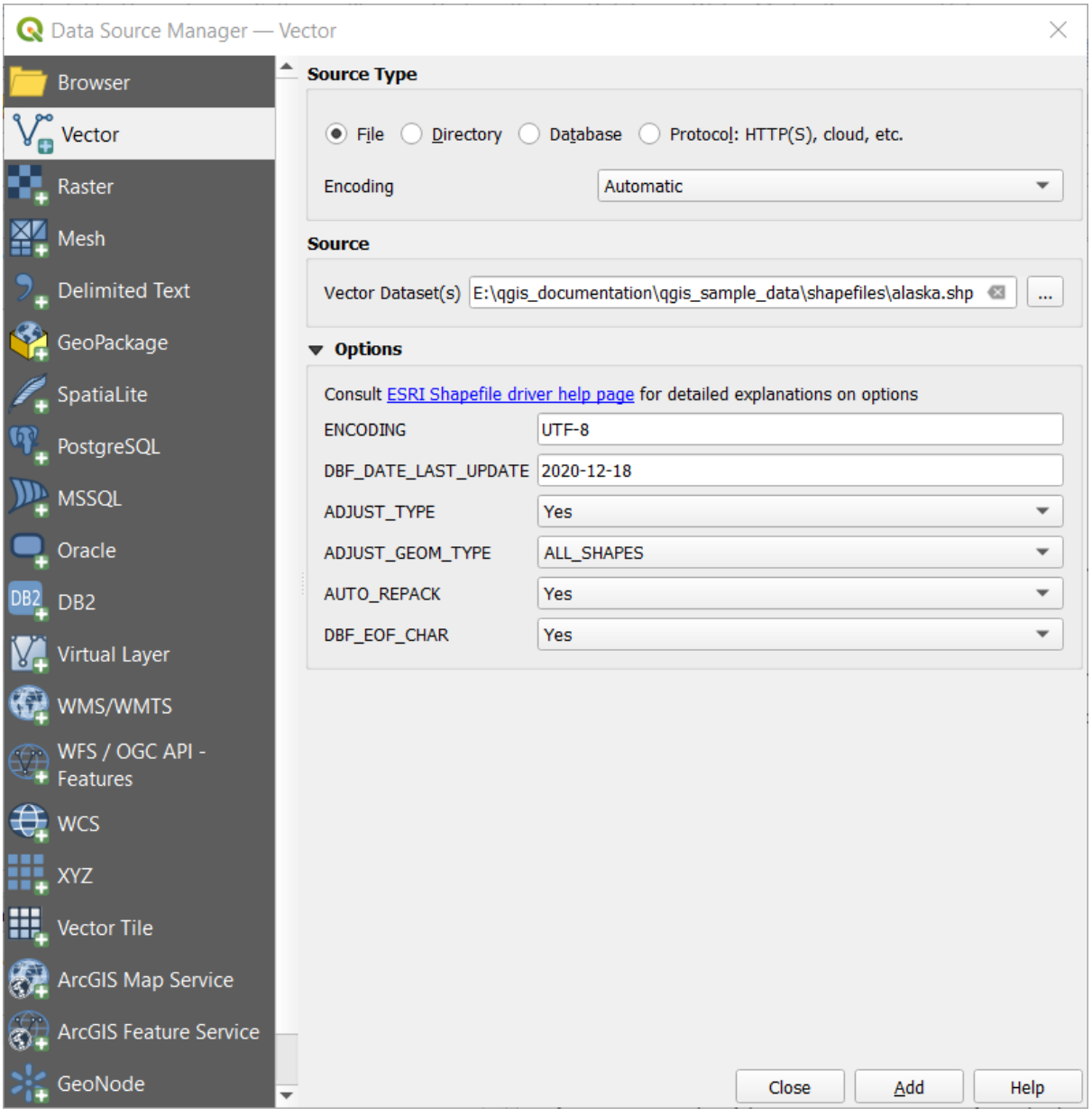


図 15.6: オプション付きでシェープファイルをロードする

6. **追加** ボタンを押すと、QGIS にファイルが読み込まれ、マップビューに表示されます。 図 15.7 は、

alaska.shp ファイルがロードされた後の QGIS を表示しています。

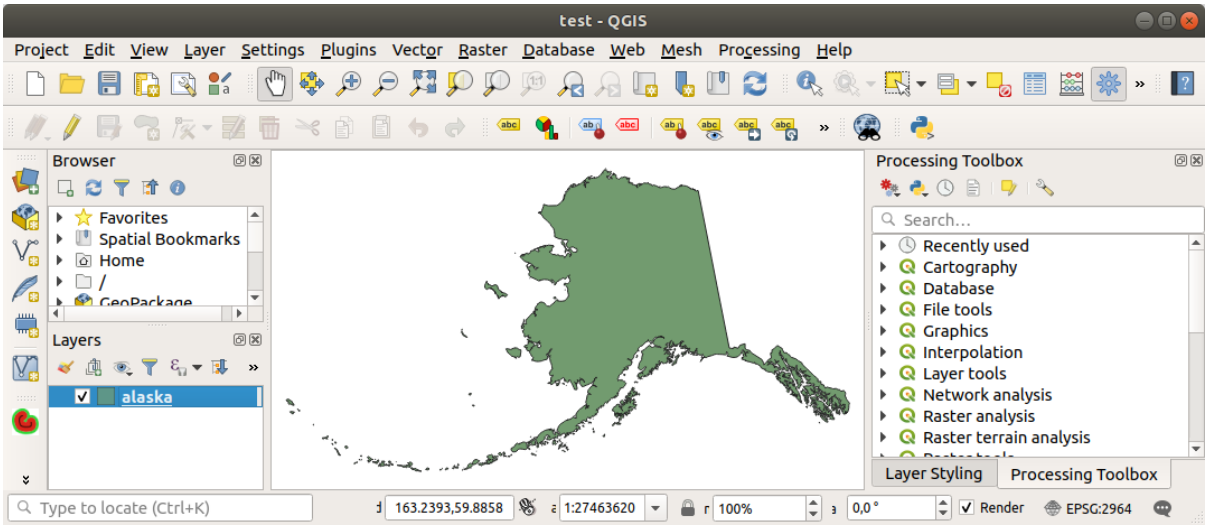


図 15.7: アラスカのシェープファイルがロードされた QGIS

注釈: GDAL ドライバはベクタファイルとラスタファイルを読み込むときにオープン動作の定義ができます。これは、ファイルが選択されたときに表示されます。オプションの詳細は https://gdal.org/drivers/vector/ および https://gdal.org/drivers/raster で説明されており、QGIS でファイルを選択すると、ハイパーリンク付きのテキストで選択したファイルタイプのドキュメントへ直接移動できます。

注釈: MapInfo （例 `.tab` ）や AutoCAD （ `.dxf` ）などいくつかのファイル形式は一つのファイル内にさまざまなジオメトリタイプを混在させることが可能なため、このようなデータセットを開く時には、レイヤごとに 1 つのジオメトリとするために、使用するジオメトリを選択するダイアログが開きます。

ベクタ と ラスタ タブでは、 **ファイル** 以外のソースタイプからレイヤを読み込むことができます：

- `ArcInfo Binary Coverage` 、 `UK. National Transfer Format` や `US Census Bureau` の raw TIGER 形式、 `OpenfileGDB` といった特定のベクタフォーマットを読み込むことができます。これらを読み込むためには、 **ソースタイプ** で **ディレクトリ** を選択します。この場合、 ... ブラウズ ボタンを押すと、ダイアログではディレクトリの選択が可能になります。
- **データベース** ソースタイプでは、既存のデータベース接続を選択したり、選択したデータベースタイプの接続を作成したりすることができます。利用可能なデータベースタイプには、 `ODBC` 、 `Esri Personal GeoDatabase` 、 `MS SQL Server` 、 `PostgreSQL` 、 `MySQL` などがあります。

  新規 ボタンを押すと、 **新しい *OGR* データベース接続を作成する** ダイアログが開きます。指定できるパラメータは **保存された接続の作成** で説明しているものと同じです。 *Open* ボタンを押すと、利用可能なテーブルの中から、例えば PostGIS が有効になっているデータベースのテーブルなどを選択することができます。
- **プロトコル** ソースタイプは、ローカル、あるいはネットワークに保存されたデータを開きます。このデータはパブリックアクセスが可能か、商用クラウドストレージサービスのプライベートバケッ

トに保存されているかのいずれかです。サポートされているプロトコルタイプは以下のとおりです：

- HTTP/HTTPS/FTP ： *URI* と、必要に応じて **認証** を指定します
- AWS S3 、 Google Cloud Storage 、 Microsoft Azure Blob 、 Alibaba OSS Cloud 、 Open Stack Swift Storage 等のクラウドストレージ。 **バケットまたはコンテナ** と **オブジェクトキー** を入力する必要があります。
- OGC WFS 3 をサポートするサービス（まだ実験段階です）。 GeoJSON または GEOJSON - Newline Delimited フォーマットを使用しているか、 CouchDB データベースに基づくものです。 *URI* が必須で、 **認証** はオプションです。
- ベクタソースタイプ全てについて、 **文字コード** の指定が可能です。 **自動** 設定を使うこともできます。

メッシュレイヤを読み込む

メッシュは通常、時間やその他の成分を持った非構造格子です。空間成分には、2D または 3D 空間の頂点、辺、面の集合が含まれます。メッシュレイヤに関する詳細な情報は **メッシュデータの操作** を参照してください。

メッシュレイヤを QGIS に追加するには、次の手順で操作します：

1. **レイヤ** メニューから選択するか、 データソースマネージャを開く ボタンをクリックして、 **データソースマネージャ** ダイアログを開きます。
2. 左のパネルの **メッシュ** タブを有効にします。
3. ... ブラウズ ボタンを押し、ファイルを選択します。 **さまざまなファイル形式** がサポートされています。
4. ファイルを選択し、 **追加** ボタンを押します。ネイティブのメッシュレンダリングを使用してレイヤが追加されます。
5. 選択したファイルが多数のメッシュレイヤを含む場合には、ダイアログが表示され、読み込むサブレイヤ選択するよう促されます。読み込むレイヤを選択して *OK* ボタンを押すと、ネイティブのメッシュレンダリングでレイヤが読み込まれます。レイヤ群をグループにして読み込むこともできます。

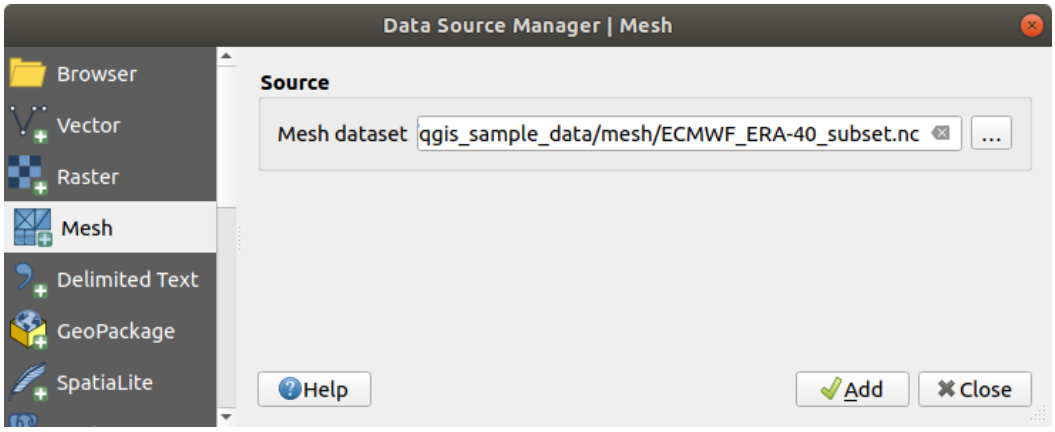


図 15.8: データソースマネージャのメッシュタブ

区切りテキストファイルをインポートする

区切りのあるテキストファイル（例えば `.txt`、`.csv`、`.dat`、`.wkt`）は上記のツールを使って読み込むことができますが、その方法ではファイルはシンプルなテーブルとして表示されます。時には、区切りのあるテキストファイルに可視化したい座標 / ジオメトリが含まれていることがあります。 *CSV* **テキストレイヤの追加** で読み込むことで、これを可視化することができます。

1. データソースマネージャを開く アイコンをクリックし、 **データソースマネージャ** ダイアログを開きます
2. *CSV* **テキスト** タブを有効にします
3. ... ブラウズ ボタンをクリックし、インポートしたい区切りのあるテキストファイル（例えば `qgis_sample_data/csv/elevp.csv`）を選択します。
4. プロジェクト内でレイヤに使用したい名前を **レイヤ名** フィールドに入力します。（例： `Elevation`）
5. 以下に説明するように、データセットや必要に合わせて設定を行います。

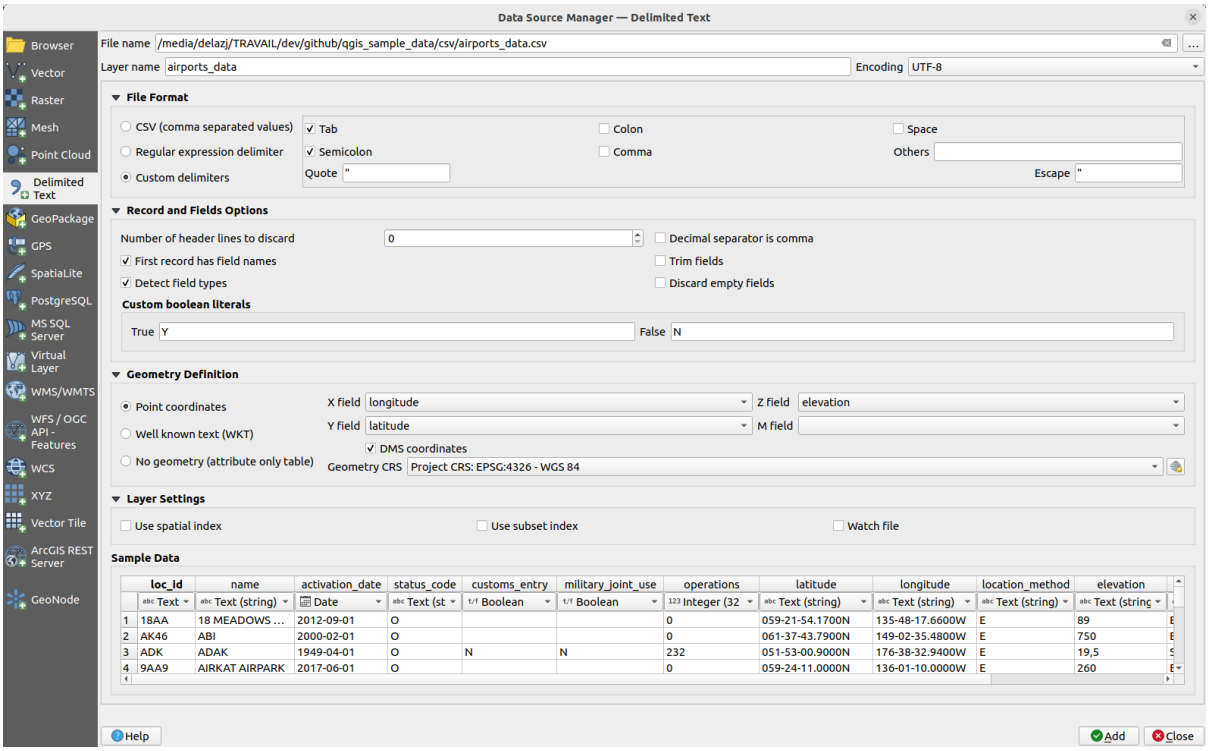


図 15.9: CSV テキストダイアログ

ファイル形式

ファイルが選択されると、QGIS は最近使用された区切り文字を使用してファイルの解析を試み、フィールドと行を識別します。QGIS がファイルを正しく解析できるようにするためには、正しい区切り文字を選択することが重要です。区切り文字には、次のいずれかを選択して指定することができます：

- *CSV*（**コンマで区切られた値**） は、コンマの文字を使います。
- **正規表現区切り** を選んだ場合は **式** フィールドにテキストを入力します。例えば、区切り文字をタブに変更するには、 `\t` （これは正規表現ではタブ文字に使用されます）を使用します。
- **カスタム区切り** は、 コンマ 、 空白 、 タブ 、 セミコロン 等の定義済み区切り文字の中から選択します。

レコードとフィールドのオプション

他にもデータ認識に使用できる便利なオプションがいくつかあります：

- **破棄するヘッダ行数** ：ファイルの始めの数行が空行あるいは別フォーマットであるときに、これをインポートしないようにする場合に便利です。
- **最初の行はフィールド名** ：最初の行の値をフィールド名として使用します。チェックがない場合には、QGIS はフィールド名に `field_1` 、 `field_2` 等を使用します。
- **フィールド型を検出する** ：フィールドの型を自動的に認識します。チェックがない場合には、全ての属性はテキストフィールドとして扱われます。
- **小数点記号にコンマを使う** ：小数点の区切り文字を強制的ににコンマとすることができます。
- **前後の空白を削除する** ：フィールドから前後の空白文字を取り除くことができます。
- **空フィールドを削除する**
- **カスタム論理値**: ブール値として検出される一組のカスタム文字列を追加することができます。

フィールド型を検出

QGIS は、（**フィールド型を検出** をチェックしている限り）フィールド型を自動的に検出しようとしますが、それは、オプションのサイドカー CSVT ファイル（参照：GeoCSV 仕様）の内容を調べ、ファイル全体をスキャンして、すべての値が実際にエラーなく変換できることを確認することによって行われます。フォールバックするときのフィールド型はテキストです。

検出されたフィールド型は、サンプルデータのプレビューテーブルのフィールド名の下に表示され、必要な場合は手動で変更できます。

次のフィールド型がサポートされています:

- ブール値 ブール値として解釈される、大文字小文字を区別しないリテラルの組みで、`1/0`, `true/false`, `t/f`, `yes/no`
- 整数 (`integer`)

- 整数 (integer - 64 bit)
- 小数点付き数値: 倍精度浮動小数点数
- 日付
- 時刻
- 日付時刻
- テキスト

ジオメトリ定義

ファイルが解析されたら、 **ジオメトリ定義** の設定を行います。

- **ポイント座標** レイヤがポイントジオメトリ型で、 *X* **属性** 、 *Y* **属性** 、 *Z* **属性** （3 次元データの場合）、 *M* **値の属性** （計測次元の場合）のフィールドを含む場合には、これを指定します。座標が度 / 分 / 秒で定義されている場合には、 **度分秒を使う** のチェックボックスにチェックを入れます。 CRS の選択 ウィジェットを使用して、適切な **ジオメトリの** *CRS* を指定します。
- *Well-known text (WKT)* オプションは、空間情報が WKT により表現されている場合に使用します：WKT ジオメトリが含まれる **ジオメトリフィールド** を選択し、 適切な **ジオメトリタイプ** を選ぶか、QGIS の自動検出に任せます。 CRS の選択 ウィジェットを使用して、 適切な **ジオメトリの** *CRS* を指定します。
- ファイルが非空間データの場合には、 **ジオメトリなし（属性のみのテーブル）** を有効にすると、通常のテーブルとして読み込まれます。

レイヤ設定

また、以下を有効にできます：

- **空間インデックスを使う** ：地物の表示や空間的な選択のパフォーマンスを改善します。
- **サブセットインデックスを使う** ：（レイヤプロパティで定義されている場合に） **サブセットフィルタ** のパフォーマンスを改善します。
- **ファイルを監視する** ：QGIS の実行中に他のアプリケーションによってファイルが変更されたかを監視します。

最後に **追加** をクリックし、マップにレイヤを追加します。この例では、 Elevation という名前のポイントレイヤがプロジェクトに追加され、QGIS の他のマップレイヤと同様に動作します。このレイヤは .csv ソースファイルに対するクエリの結果であり（従ってこれにリンクされているため）、ディスク上に空間レイヤを取得するには **保存する** 必要があります。

サンプルデータ

パーサーのプロパティを設定すると、サンプルデータのプレビューが適用された設定に合わせて更新されます。

また、サンプルデータテーブルでは、自動的に決定されたカラムの型を上書きすることもできます。

DXF ファイルや DWG ファイルをインポートする

DXF ファイルや DWG ファイルはブラウザパネルからドラッグ＆ドロップするだけで QGIS に追加できます。プロジェクトに追加したいサブレイヤを選択するプロンプトが表示されます。レイヤはランダムなスタイルのプロパティで追加されます。

注釈: いくつかのジオメトリタイプ（ポイント、ライン、ポリゴン）を含む DXF ファイルは、*<filename.dxf> entities <geometry type>* のような名前でレイヤが生成されます。

---

dxf/dwg ファイルの構成とシンボロジを QGIS でも保つためには、 **プロジェクト　インポートとエクスポート　*DWG/DXF* からレイヤをインポート**... から専用のツールを使用するとよいでしょう。このツールにより、次の操作を行います：

1. 図面ファイルから GeoPackage データベースに要素をインポートする
2. インポートした要素をプロジェクトに追加する

DWG/DXF **のインポート** ダイアログでは、図面ファイルの内容をインポートするために以下の情報を入力する必要があります：

1. **ターゲットパッケージ**、つまりデータが保存される新規 GeoPackage ファイルの場所を入力します。既存のファイルを指定した場合には、ファイルは上書きされます。
2. 図面ファイルのデータの座標参照系を指定します。
3. ☑ **ブロック参照を展開する** をチェックすると、図面ファイル内のブロックを標準的な要素としてインポートします。
4. ☑ **カーブを使用する** をチェックすると、インポートされたレイヤを カーブ ジオメトリタイプに変換します。
5. **インポート** ボタンを使用して、使用する DWG/DXF ファイル（1 つの GeoPackage に 1 ファイル）を選択します。図面ファイルの内容が GeoPackage データベースに自動的に追加されます。ファイルのサイズにもよりますが、これには時間がかかる場合があります。

`.dwg` ファイルまたは `.dxf` ファイルのデータが GeoPackage データベースへとインポートされたら、ダイアログの下半分にあるフレームにインポートファイルからのレイヤのリストが表示されます。ここから QGIS プロジェクトに追加したいレイヤを選択できます：

1. フレーム上部で、プロジェクト内で図面ファイルをグループ化するための **グループ名** を設定します。
2. 表示するレイヤにチェックを入れます。選択された各レイヤは、図面レイヤのポイント、ライン、ラベル、面地物に対するベクタレイヤを含むアドホックなグループに追加されます。レイヤのスタイルは、元々 *CAD で持っていたスタイルと似た見た目になります。

3. 最初からレイヤが表示されているかどうかを選択します。
4. **レイヤを結合する** オプションをチェックすると、全てのレイヤを単一のグループにまとめます。
5. *OK* ボタンを押すと、QGIS でレイヤを開きます。

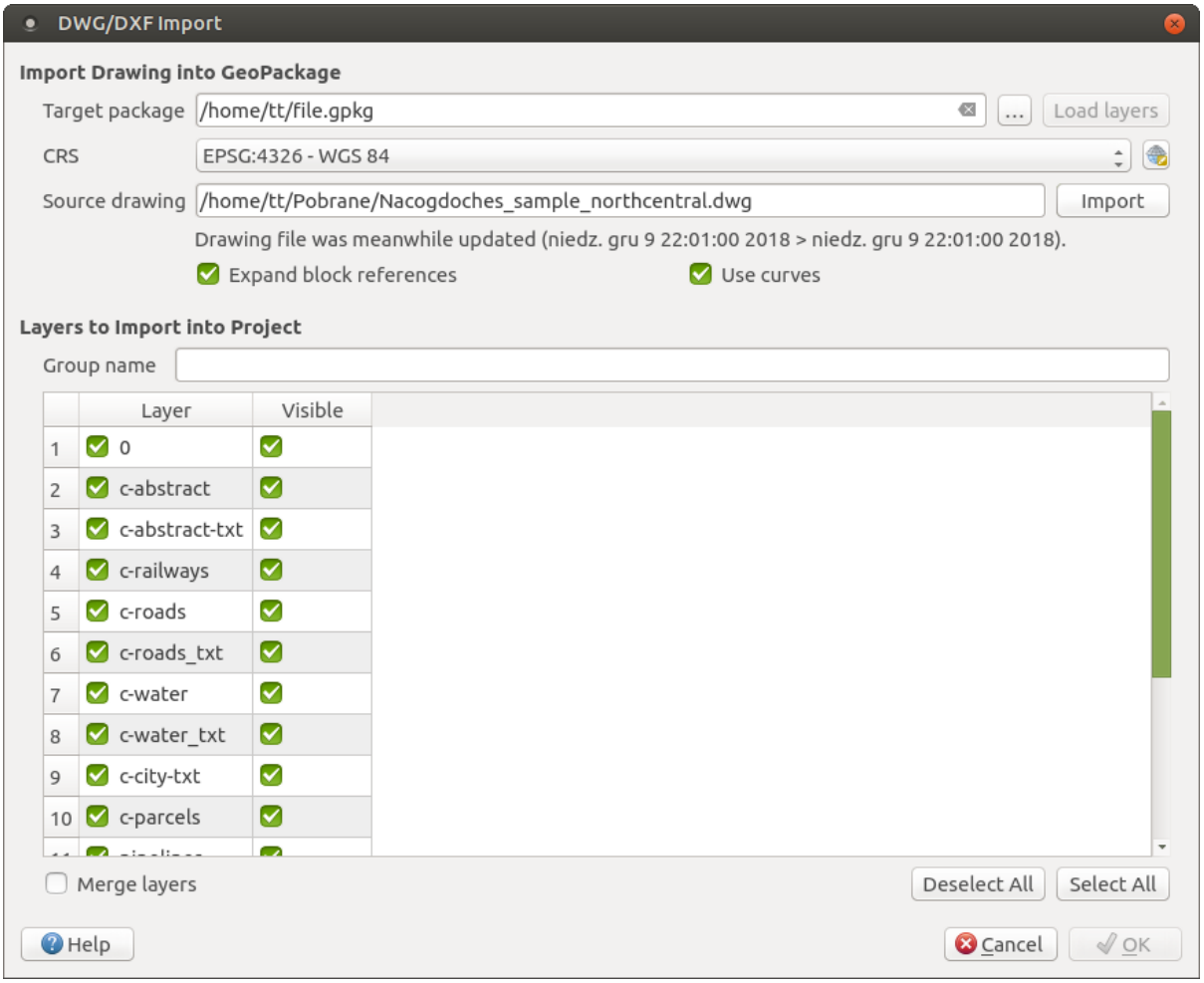


図 15.10: DWG/DXF ファイルのインポートダイアログ

OpenStreetMap ベクタのインポート

多くの国ではデジタル道路地図等のジオデータが無料で利用できないため、OpenStreetMap プロジェクトは人気があります。OSM プロジェクトの目的は、GPS データや航空写真、ローカルな知識からフリーで編集可能な世界地図を作成することです。この目的を支援するため、QGIS は OSM データのサポートを提供しています。

ブラウザパネル を使用して `.osm` ファイルをマップキャンバスに読み込むと、ジオメトリタイプに基づいてサブレイヤを選択するためのダイアログが現れます。読み込んだレイヤには `.osm` ファイル内のそのジオメトリタイプの全てのデータが含まれており、 `osm` ファイルのデータ構造が保持されています。

SpatiaLite レイヤ

初めて SpatiaLite データベースからデータをロードするときは、始めに以下の操作を行います：

- SpatiaLite レイヤの追加 ツールバーボタンをクリックする
- **レイヤ** **レイヤの追加** メニューから *SpatiaLite* **レイヤの追加**... オプションを選択する
- `Ctrl+Shift+L` を押す

これによりウィンドウが現れ、QGIS が既に知っている SpatiaLite データベース（ドロップダウンメニューから選択）に接続するか、新しいデータベースへの新規接続を定義します。新規接続を定義するには、 **新規** をクリックし、ファイルブラウザを使用して SpatiaLite データベースを指定します。SpatiaLite データベースは `.sqlite` 拡張子を持つファイルです。

また、QGIS は SpatiaLite で編集可能なビューもサポートしています。

GPS

GPS データを保存するためのファイル形式は何十種類もあります。QGIS が使用している形式は GPX（GPS eXchange format）と呼ばれ、同じファイル内に複数のウェイポイント、ルート、トラックを含めることができる標準的な交換形式です。

... ブラウズ ボタンを使用して GPX ファイルを選択し、GPX ファイルから読み込みたい地物タイプをチェックボックスを使用して選択します。各地物タイプは別々のレイヤで読み込まれます。

GPS データの操作についての詳細は *GPS* **データの操作** を参照。

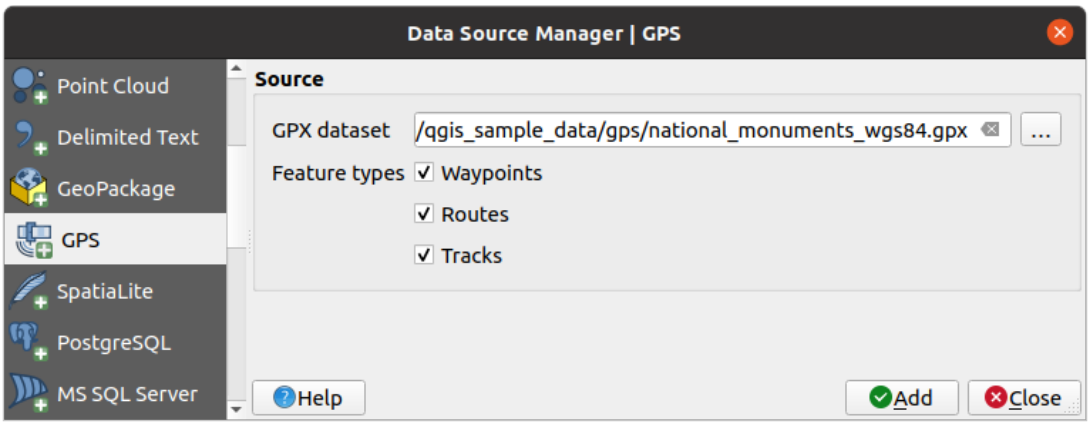


図 15.11: GPS データの読み込みダイアログ

GRASS

GRASS ベクタデータでの作業は *GRASS GIS* **の統合** セクションで説明しています。

データベース関連ツール

保存された接続の作成

QGIS がサポートするデータベース形式からテーブルを読み書きするためには、そのデータベースへの接続を作成する必要があります。 *QGIS* **ブラウザパネル** はデータベースへ接続し使用するための最もシンプルで推奨される方法ですが、QGIS には各データベースに接続しテーブルをロードするための別のツールも用意されています。

-  *PostGIS* **レイヤの追加**... またはキーボードで `Ctrl+Shift+D`
-  *MS SQL* **サーバーレイヤを追加**
-  *Oracle Spatial* **レイヤの追加**... またはキーボードで `Ctrl+Shift+O`
-  *SAP HANA* **空間レイヤを追加**... またはキーボードで `Ctrl+Shift+G`

これらのツールには **レイヤ管理ツールバー** や **レイヤ** **レイヤの追加** メニューからアクセスすることができます。SpatiaLite データベースへの接続は *SpatiaLite* **レイヤ** で説明しています。

Tip: QGIS ブラウザパネルからデータベースへの接続を作成する

ブラウザツリー内で対応するデータベース形式を選択し、右クリックして接続を選択すると、データベース接続ダイアログが開きます。

ほとんどの接続ダイアログは、PostgreSQL データベースへの接続ツールを例として以下で説明する共通の基本事項に従っています。他のプロバイダに特有の追加設定については、対応する説明を以下で行っています：

- *MS SQL* **サーバーへの接続**;
- *Oracle Spatial* **のへの接続**;
- *SAP HANA* **への接続**.

PostGIS データソースを初めて使用する時には、データを含むデータベースへの接続を作成しなければなりません。まずは上記のように適切なボタンを押して、 *PostGIS* **テーブルの追加** ダイアログ（ 図 15.14 参照）を開きます。接続マネージャにアクセスするには、 **新規** ボタンをクリックして **新規** *PostGIS* **接続を作成** ダイアログを表示します。

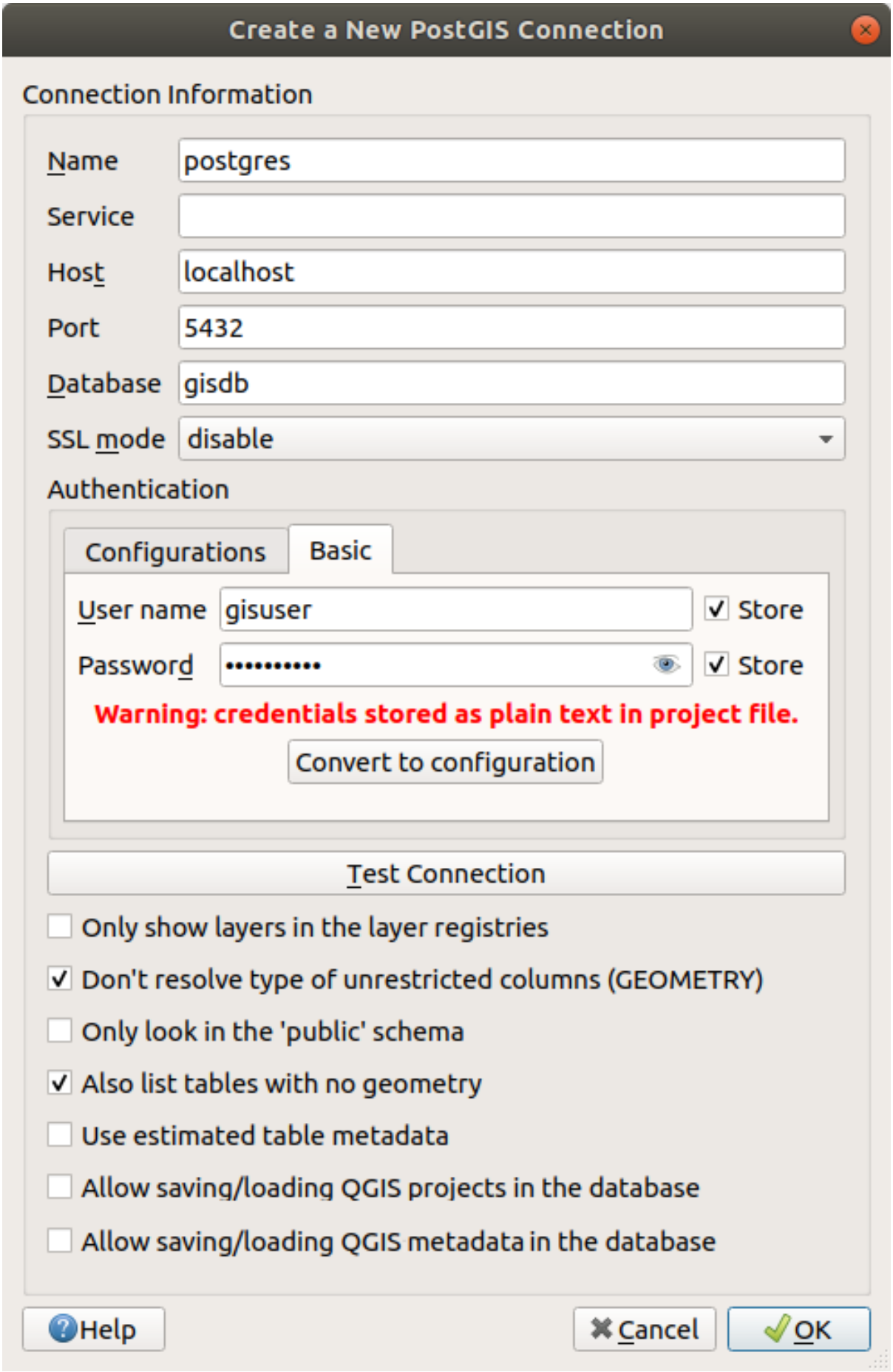


図 15.12: 「新規 PostGIS 接続を作成」ダイアログ

PostGIS の接続に必要なパラメータを以下に説明します。他のタイプのデータベースについては **特定の接続要件** を参照し、違いを確認してください。

- **名前** ：接続の名前です。これは **データベース** の名前と同じです。
- **サービス** ：ホスト名/ポート番号（場合によってはデータベース名）の代わりに使用されるサービスパラメータです。これは `pg_service.conf` ファイルで定義されます。詳細については *PostgreSQL* **サービス接続ファイル** のセクションをチェックしてください。
- **ホスト** ：データベースのホスト名です。これは、TCP/IP 接続を開いたり、ホストを ping したりするために使用されるような、解決可能なホスト名でなければなりません。データベースが QGIS と同じコンピュータ上にある場合は、ここには単に *localhost* と入力します。
- **ポート** ： PostgreSQL データベースサーバーがリッスンするポート番号です。PostGIS のデフォルトのポートは `5432` です。
- **データベース** ：データベース名です。
- *SSL* **モード** ：SSL 暗号化の設定です。以下の選択肢があります：
  - *Prefer*（デフォルト）：セキュリティはどうでもよいが、サーバがそれをサポートするのであれば暗号化のオーバヘッドを払ってもよい
  - *Require* ：データを暗号化して欲しい。そしてオーバヘッドも受け入れる。サーバに接続する必要時に常にネットワークが確認してくれることを信用する
  - *Verify CA* ：データを暗号化して欲しい。そしてオーバヘッドも受け入れる。信頼するサーバに確実に接続したい
  - *Verify Full* ：データを暗号化して欲しい。そしてオーバヘッドも受け入れる。信頼するサーバに接続すること、そのサーバが指定したものであることを確実にしたい
  - *Allow* ：セキュリティはどうでもよいが、サーバがそれを強く要求するのであれば暗号化のオーバヘッドを払ってもよい
  - *Disable* ：セキュリティはどうでもよく、暗号化の負荷を払いたくない
- **認証** ベーシック
  - **ユーザー名** ：データベースのログインに使用されるユーザー名です。
  - **パスワード** ：データベースの接続に **ユーザー名** とともに使用するパスワードです。

  ユーザー名 と パスワード のパラメータはどちらか一方もしくは両方を保存することができます。その場合はこのデータベースに接続するたびにデフォルトで使用されます。保存されていない場合は、次回の QGIS セッションでデータベースに接続するための認証情報を入力するように促されます。入力した接続パラメータは一時的な内部キャッシュに保存され、現在の QGIS セッションが終了するまで、同じデータベースのユーザ名/パスワードが要求されるたびに内部キャッシュから返されます。

> **警告: QGIS ユーザー設定とセキュリティ**
>
> **認証** タブで **ユーザー名** と **パスワード** を保存すると、保護されていない認証情報が接続設定に保持されます。 例えば、プロジェクトファイルを他者と共有した場合には、これらの **認証情報が見えてしまいます** 。従って、代わりに **認証設定** の中に資格情報を保存する（ **設定** タブを使

> 用します。詳細は **認証システム** を参照）か、サービス接続ファイル（例は *PostgreSQL* **サービス接続ファイル** 参照）内に資格情報を保存することをお勧めします。

- **認証** 設定タブでは、認証設定を選択します。 ボタンを押して、設定を追加できます。選択肢には以下のものがあります：
  - ベーシック認証
  - PKI PKCS#12 認証
  - PKI パス認証
  - PKI stored identity certificate

オプションで、データベースのタイプに応じて以下のチェックボックスをアクティブにできます:

- **レイヤレジストリ内のレイヤのみ表示する**
- **制限のないカラムの型を解決しない（*GEOMETRY*）**
- *'public'* **スキーマのみ参照する**
- **ジオメトリを持たないテーブルもリストする**
- **概算されたテーブルメタデータを利用する**
- *QGIS* **プロジェクトのデータベース保存・読み込みを許可する** - 詳細については **ここ** を参照
- *QGIS* **プロジェクトのデータベース保存・読み込みを許可する** - 詳細は **ここ**

Tip: 概算されたテーブルメタデータを使用して作業をスピードアップする

レイヤを初期化する際に、データベーステーブルに格納されているジオメトリの特性をはっきりさせるために様々なクエリが必要になることがあります。 **概算されたテーブルメタデータを利用する** オプションがチェックされている場合、これらのクエリはテーブル全体ではなく行のサンプルのみを調べてテーブルの統計情報に使用します。これにより大規模なデータセットに対する操作を劇的に高速化することができますが、レイヤの特性が不正確（例えばフィルタされたレイヤの地物カウントが正確に決定できない）となり、さらには実際にはユニークであるはずの列がユニークにならないなどの奇妙な動作を引き起こす場合があります。

全てのパラメータとオプションを設定したら、**接続テスト** ボタンをクリックして接続をテストするか、*OK* ボタンをクリックして設定を適用します。それからは *PostGIS* **テーブルの追加** から **接続** をクリックすると、選択したデータベースのテーブルが並んだダイアログが（ 図 15.14 に示すように）表示されます。

特定の接続要件

データベースタイプの特殊性のため、提供されるオプションは同じではありません。データベース固有のオプションを以下で説明します。

PostgreSQL サービス接続ファイル

サービス接続ファイルは、PostgreSQL の接続パラメータを単一のサービス名と関連付けできます。そのサービス名をクライアントで指定すると、関連付けられた設定が使用されます。

これは、*nix システム（GNU/Linux、MacOS など）では `.pg_service.conf`、Windows では `pg_service.conf` と呼ばれています。

サービスファイルは以下のようなものです:

```
[water_service]
host=192.168.0.45
port=5433
dbname=gisdb
user=paul
password=paulspass

[wastewater_service]
host=dbserver.com
dbname=water
user=waterpass
```

注釈: 上の例には、 `water_service` と `wastewater_service` の 2 つのサービスがあります。 接続したいサービスの名前（囲みブラケットは除く）を指定するだけで、QGIS や pgAdmin 等からサービスに接続することができます。 `psql` でこのサービスを使用したい場合には、psql コマンドを使用する前に `export PGSERVICE=water_service` のような操作を行う必要があります。

ここ では、全ての PostgreSQL のパラメータを確認できます。

注釈: サービスファイル内にパスワードを保存したくない場合には、 .pg_pass オプションを使用します。

---

*nix オペレーティングシステム（GNU/Linux、macOS 等）では、 `.pg_service.conf` ファイルはユーザーのホームディレクトリに保存することができ、PostgreSQL クライアントはこれを自動的に見つけます。例えば、ログインユーザーが `web` ならば、（他の環境変数を何も指定しなくとも）そのままでうまくいくためには、 `.pg_service.conf` は `/home/web/` ディレクトリに保存されている必要があります。

`PGSERVICEFILE` 環境変数を作成することで、サービスファイルの場所を指定することができます（例えば、*nix OS では `export PGSERVICEFILE=/home/web/.pg_service.conf` コマンドを実行して、 `PGSERVICEFILE` 変数を一時的に設定します）。

また、.pg_service.conf ファイルを pg_config --sysconfdir が指す場所に配置するか、PGSYSCONFDIR 環境変数を追加してサービスファイルが含まれるディレクトリを指定することで、サービスファイルをシステム全体（全てのユーザー）で利用できるようにすることもできます。ユーザーファイルとシステムファイルで同じ名前のサービス定義が存在する場合は、ユーザーファイルが優先されます。

警告: Windows ではいくつか注意点があります。

- サービスファイルは .pg_service.conf ではなく、 pg_service.conf という名前で保存する必要があります。
- サービスファイルが機能するためには、Unix のフォーマットで保存する必要があります。それを行うための方法の 1 つは、 Notepad++ でサービスファイルを開き **編集** **改行コード変換** *UNIX（LF）* **ファイル保存** とします。
- 環境変数はさまざまな方法で追加することができます。確実にうまくいくと知られているテスト済みの方法は、**コントロールパネル** **システムとセキュリティ** **システム** **システムの詳細設定** **環境変数** と操作して、PGSERVICEFILE 変数にパス、例えば C:\Users\John\pg_service.conf を設定して追加します。
- 環境変数を追加した後は、コンピュータを再起動する必要もあるかもしれません。

Oracle Spatial のへの接続

Oracle Spatial の空間機能は、ユーザーが地理的データや位置情報データを Oracle データベース内のネイティブ型で管理するのに役立ちます。 **保存された接続の作成** のいくつかのオプションに加えて、接続ダイアログでは下記の設定を行います。

- **データベース** :Oracle インスタンスの SID または SERVICE_NAME.
- **ポート**: Oracle データベースサーバーがリッスンしているポート番号。デフォルトのポートは 1521 です。
- **オプション** ：Oracle 接続特有のオプション（例えば OCI_ATTR_PREFETCH_ROWS, OCI_ATTR_PREFETCH_MEMORY など）。オプション文字列はセミコロン区切りで、オプション名またはオプション=値 のペアの形式です。
- **ワークスペース**：切り替え先のワークスペースを指定します。
- **スキーマ** ：データが格納されているスキーマです。

オプションで、以下のチェックボックスを有効にできます：

- ☑ **メタデータテーブルのみを調べる** ：表示されるテーブルを all_sdo_geom_metadata ビュー内にあるものに制限します。これは、空間テーブルの初期表示を高速化することができます。
- ☑ **ユーザのテーブルのみを参照する** ：空間テーブルを検索するときに、検索をユーザが所有するテーブルに制限します。
- ☑ **ジオメトリを持たないテーブルもリストする** ：ジオメトリを持たないテーブルもデフォルトでリストするように指定します。

- **概算されたテーブルメタデータを利用する** ：レイヤが読み込まれると、Oracle テーブルのさまざまなメタデータが必要になります。これには、テーブルの行数、ジオメトリタイプ、ジオメトリ列のデータの空間的広がりなどの情報が含まれます。テーブルに多数の行が含まれていると、このメタデータの決定に時間がかかることがあります。このオプションを有効にすると、以下のような高速なテーブルメタデータ操作が行われます。行数カウントは `all_tables.num_rows` から決定されます。テーブルの空間範囲は、レイヤフィルタが適用されている場合でも、常に SDO_TUNE.EXTENTS_OF 関数で決定されます。テーブルのジオメトリは、テーブル内の最初の 100 行の非ヌルなジオメトリから決定されます。
- **存在するジオメトリタイプのみ** ：存在しているジオメトリタイプのみをリストし、他のジオメトリタイプは追加しません。
- **追加のジオメトリ属性を含める** 。

Tip: Oracle Spatial レイヤ

通常、ORACLE Spatial レイヤは **USER_SDO_METADATA** テーブルのエントリで定義されます。

選択ツールが正しく動作するようにするためには、テーブルに **主キー** があることをお勧めします。

MS SQL サーバーへの接続

MS SQL サーバー接続の新規作成ダイアログでは、**保存された接続の作成** のいくつかのオプションに加えて、**プロバイダ/DSN** 名を入力することができます。利用可能なデータベースを表示することもできます。

SAP HANA への接続

注釈: SAP HANA データベースに接続するためには、SAP HANA Client が必要となります。自分のプラットフォーム向けの SAP HANA Client は、 SAP Development Tools website からダウンロードできます。

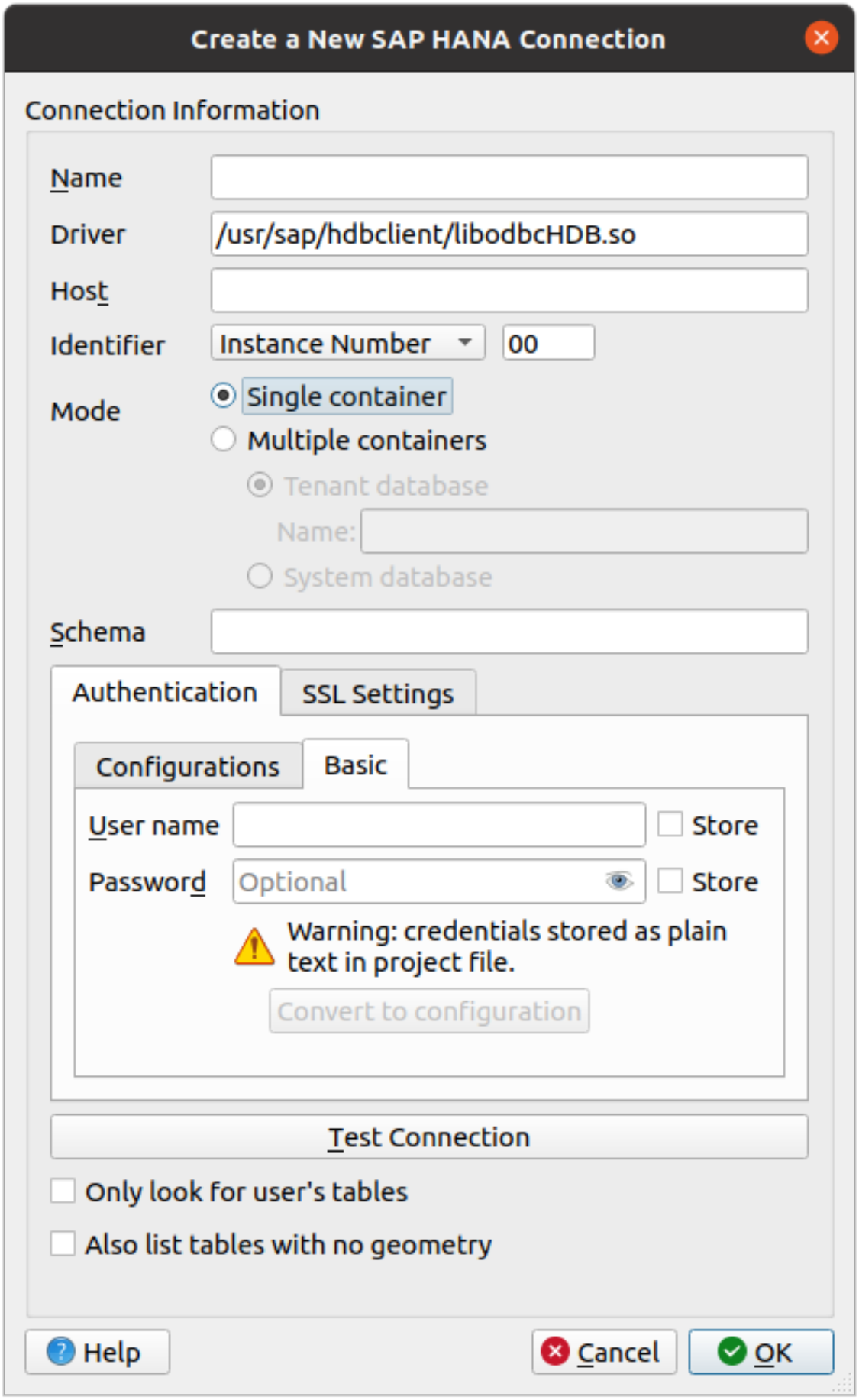


図 15.13: 「新規 SAP HANA 接続を作成」ダイアログ

以下のパラメータを入力します：

- **名前** ：コネクションの名前
- **ドライバー** ：HANA ODBC ドライバーの名前。QGIS の 64 ビット版を使用している場合は `HDBODBC` 、32 ビット版を使用している場合は `HDBODBC32` です。適切なドライバー名は自動的に入力されます。
- **ドライバー** ：SAP HANA ODBC ドライバーが `/etc/odbcinst.ini` 内に登録されている名前、または SAP HANA ODBC ドライバーへのフルパスのいずれかを入力します。SAP HANA Client のインストーラは、デフォルトで ODBC ドライバーを `/usr/sap/hdbclient/libodbcHDB.so` にインストールします。
- **ホスト** ：データベースのホスト名
- **識別子** ：ホスト上で接続するインスタンスを識別します。これは、 **インスタンス番号** または **ポート番号** のどちらかです。インスタンス番号は 2 桁の数字で、ポート番号は 1 から 65,535 の範囲です。
- **モード** ：SAP HANA インスタンスの実行モードを指定します。この設定は、 **識別子** を **インスタンス番号** に設定した場合にのみ考慮されます。データベースがマルチプルコンテナをホストしている場合は、 **テナントデータベース** に指定した名前のテナントや、システムデータベースに接続できます。
- **スキーマ** ：このパラメータはオプションです。スキーマ名を指定した場合、QGIS はそのスキーマのデータのみを検索します。フィールドを空のままとした場合には、QGIS はすべてのスキーマのデータを検索します。
- **認証　ベーシック**
  - **ユーザー名** ：データベースへの接続に使用するユーザー名
  - **パスワード** ：データベースへの接続に使用するパスワード
- *SSL* **設定**
  - *TLS/SSL* **暗号化** ：TLS 1.1 - TLS1.2 暗号化を有効にします。サーバーは利用可能な中で上位のものを選択します。
  - **プロバイダ** ：SSL 通信に使用する暗号化ライブラリのプロバイダを指定します。 *sapcrypto* はすべてのプラットフォームで動作しますが、 *openssl* は で、 *mscrypto* は で動作し、 *commoncrypto* は CommonCryptoLib がインストールされている必要があります。
  - *SSL* **証明を有効化** ：チェックした場合、 **公開鍵を含む鍵ファイルを信頼する** に指定したトラストストアを使用して SSL 証明が有効化されます。
  - **証明書のホスト名を上書き** ：サーバーの身元確認に使用するホスト名を指定します。ここで指定するホスト名は、接続を確立したホスト名ではなく、サーバーの身元を確認するものです。ホスト名に * を指定した場合、サーバーのホスト名は検証されません。その他のワイルドカードは使えません。
  - **秘密鍵を含む鍵ファイル** ：現在は無視されます。このパラメータによって、将来的にはユーザ名とパスワードによる認証の代わりに、証明書による認証ができるようになるかもしれません。
  - **公開鍵を含む鍵ファイルを信頼する** ：OpenSSL を使用する場合に、サーバーの公開鍵証明を含むトラストストアファイルへのパスを指定します。通常は、トラストストアにはルート証明書またはサーバーの公開証明書に署名した認証局の証明書が含まれます。暗号化ライブラリに CommonCryptoLib や msCrypto を使用する場合には、このプロパティは空のままにしておきます。

- **ユーザーのテーブルのみ参照する**：チェックを入れた場合、QGIS はデータベースに接続したユーザーが所有するテーブルとビューのみを検索します。
- **ジオメトリを持たないテーブルもリストする** チェックを入れた場合、QGIS は空間カラムを持たないテーブルやビューも検索します。

Tip: SAP HANA Cloud への接続

SAP HANA Cloud インスタンスへ接続したい場合には、通常は **ポート番号** を 443 として、 *TLS/SSL* **暗号化** にチェックを入れる必要があります。

データベースのレイヤを読み込む

データベースに 1 つ以上の接続が定義できたら（ **保存された接続の作成** を参照）、そこからレイヤを読み込むことができます。もちろん、このためにはデータが利用可能なものである必要があります。PostGIS データベースへのデータのインポートについての議論は、 *PostgreSQL* **へデータをインポートする** のセクションを参照してください。

データベースからレイヤを読み込むには、以下の手順に従います：

1. " <database> テーブルの追加" ダイアログを開きます（ **保存された接続の作成** 参照）。
2. 接続をドロップダウンリストから選択し、 **接続** ボタンをクリックします。
3. **ジオメトリを持たないテーブルもリストする** を選択または非選択にできます
4. オプションとして、いくつかの **検索オプション** を使用して、検索にマッチしたもののみにテーブルのリストを削減できます。このオプションを **接続** ボタンを押す前に設定することもでき、データベースからのテーブル取得を高速化することができます。
5. 利用可能なレイヤのリスト内で、追加したいレイヤを探してください。
6. 追加したいレイヤをクリックして選択します。 `Shift` キーまたは `Ctrl` キーを押しながらクリックすることで、複数のレイヤを選択できます。
7. 利用可能ならば、 **フィルタの設定** ボタン（もしくはレイヤをダブルクリック）を使用して **クエリビルダ** ダイアログ（ **クエリビルダ** のセクション参照）を開始し、選択したレイヤからどの地物を読み込むかを定義します。フィルタ式はリストの `sql` 列に表示されます。この制約条件は、 **レイヤプロパティ　ソース　プロバイダ地物フィルタ** フレームで削除または編集ができます。
8. `id` で選択 カラムのチェックボックスはデフォルトで有効になっていますが、これは属性なしの地物 id を取得し、たいていの場合でデータの読み込みを高速化します。
9. **追加** ボタンをクリックし、マップにレイヤを追加します。

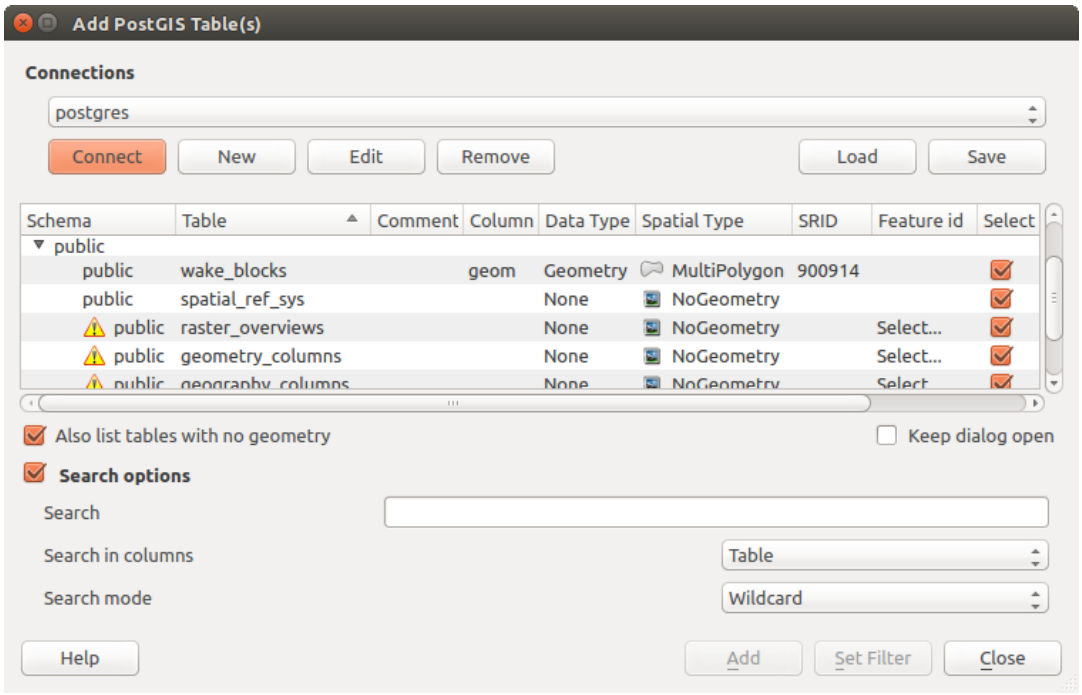


図 15.14: PostGIS テーブルを追加ダイアログ

Tip: ブラウザパネルを使用してデータベースのテーブル読み込みを高速化する

データソースマネージャ からデータベーステーブルを追加しようとすると、QGIS が各テーブルについて統計量やプロパティ（例えば、ジオメトリタイプやフィールド、CRS、地物の数など）をあらかじめ取得するため、時間がかかることがあります。これを避けるには、 **接続の設定** ができたら、 **ブラウザパネル** または *DB* **マネージャ** を使用して、データベーステーブルをマップキャンバスへドラッグ＆ドロップする方法が良いでしょう。

15.1.4 レイヤメタデータ検索パネル

レイヤメタデータ検索パネルでは、登録されているメタデータプロバイダからレイヤメタデータを参照し、プロジェクトに追加することができます。

リストは、テキスト、現在のプロジェクト、マップキャンバスの範囲によってフィルタリングすることができます。

メタデータのソースは、プラグインによって拡張可能なレイヤメタデータプロバイダのシステムによって実装されています。

QGIS は、メタデータの保存を可能にする接続からメタデータを取得するレイヤメタデータプロバイダをすぐに提供します（詳しくは **メタデータをデータベースに保存する**）。

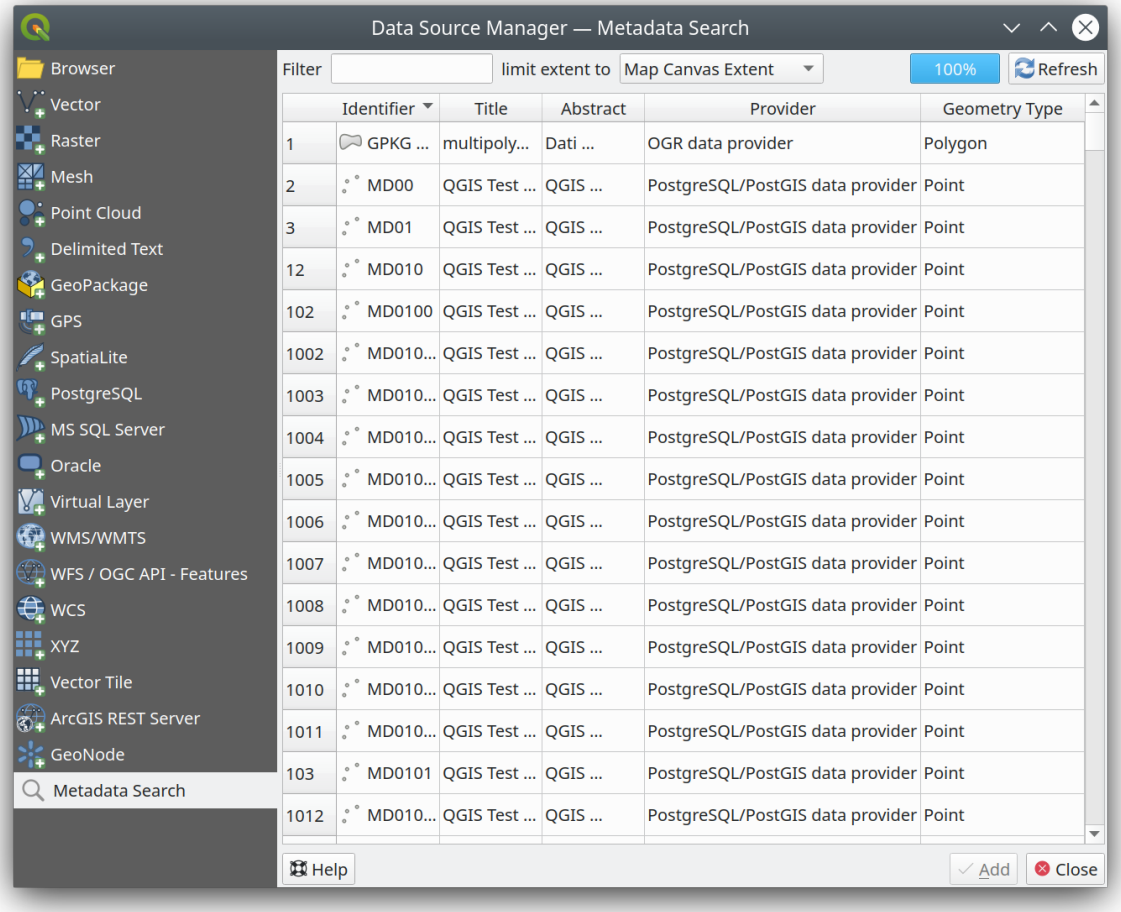


図 15.15: レイヤメタデータ検索パネル

15.1.5 QGIS カスタム形式

QGIS には 2 つのカスタム形式があります：

- 一時スクラッチレイヤ：プロジェクトに結び付いたメモリレイヤです（詳細は **新しい一時スクラッチレイヤを作成する** を参照してください）
- 仮想レイヤ：他のレイヤに対するクエリの結果として得られるレイヤです（詳細は **仮想レイヤを作成する** を参照してください）

15.1.6 QLR - QGIS レイヤ定義ファイル

レイヤのコンテキストメニューで **エクスポート　レイヤ定義ファイルとして保存**... を使用して、レイヤの定義を **レイヤ定義ファイル** （QLR - `.qlr` ）として保存することができます。

QLR 形式によって、 "完全な" QGIS レイヤを他の QGIS ユーザーと共有することができます。 QLR ファイルには、データソースへのリンクとレイヤのスタイル設定に必要な全ての QGIS スタイル情報が含まれています。

QLR ファイルはブラウザパネル内に表示され、レイヤを（保存されたスタイルで）レイヤパネルに追加するために使用できます。QLR ファイルをシステムのファイルマネージャからマップキャンバスへとドラッグ＆ドロップすることもできます。

15.1.7 ウェブサービスへ接続する

QGIS を使用すると、さまざまな種類の OGC ウェブサービス（WM(T)S、WFS(-T)、WCS、CSW、...）にアクセスすることができます。QGIS Server のおかげで、このようなサービスを公開することもできます。QGIS-Server-manual には、これらの機能についての説明があります。

ベクタタイルサービスを利用する

ベクタタイルサービスは、 **データソースマネージャ** ダイアログの **ベクタタイル** タブまたは **ブラウザ** パネルの *Vector Tiles* エントリのコンテキストメニューから追加することができます。サービスは、 **新規一般接続**... か **新規** *ArcGIS Vector Tile Service* **接続**... のいずれかです。

以下を入力することで、サービスを設定できます：

- **名前**
- *URL* ：一般接続の場合は `http://example.com/{z}/{x}/{y}.pbf` のような形式、ArcGIS ベースのサービスの場合は `http://example.com/arcgis/rest/services/Layer/VectorTileServer` のような形式です。サービスはタイルを `.pbf` 形式で提供している必要があります。
- **最小ズームレベル** および **最大ズームレベル** ：ベクタタイルはピラミッド構造を持っています。これらのオプションを使用すると、タイルピラミッドから個別にレイヤを生成することができます。これらのレイヤは、QGIS でベクタタイルのレンダリングに使用されます。

  メルカトル図法（OpenStreetMap Vector Tiles で使用）の場合、ズームレベル 0 は世界全体を 1:500.000.000 の縮尺で表現します。ズームレベル 14 は、縮尺 1:35.000 での表現です。
- *Style URL* ：MapBox GL JSON スタイル設定への URL です。これを設定すると、この接続からのレイヤが QGIS に追加される場合に、常にそのスタイルが適用されます。ArcGIS ベクタタイルサービス接続の場合、この URL はサーバー側の設定で指定されるデフォルトのスタイル設定を上書きします。
- **認証** ：必要に応じて設定
- **リファラー**

図 15.16 は、MapTiler planet Vector Tiles サービスの設定ダイアログを示しています。

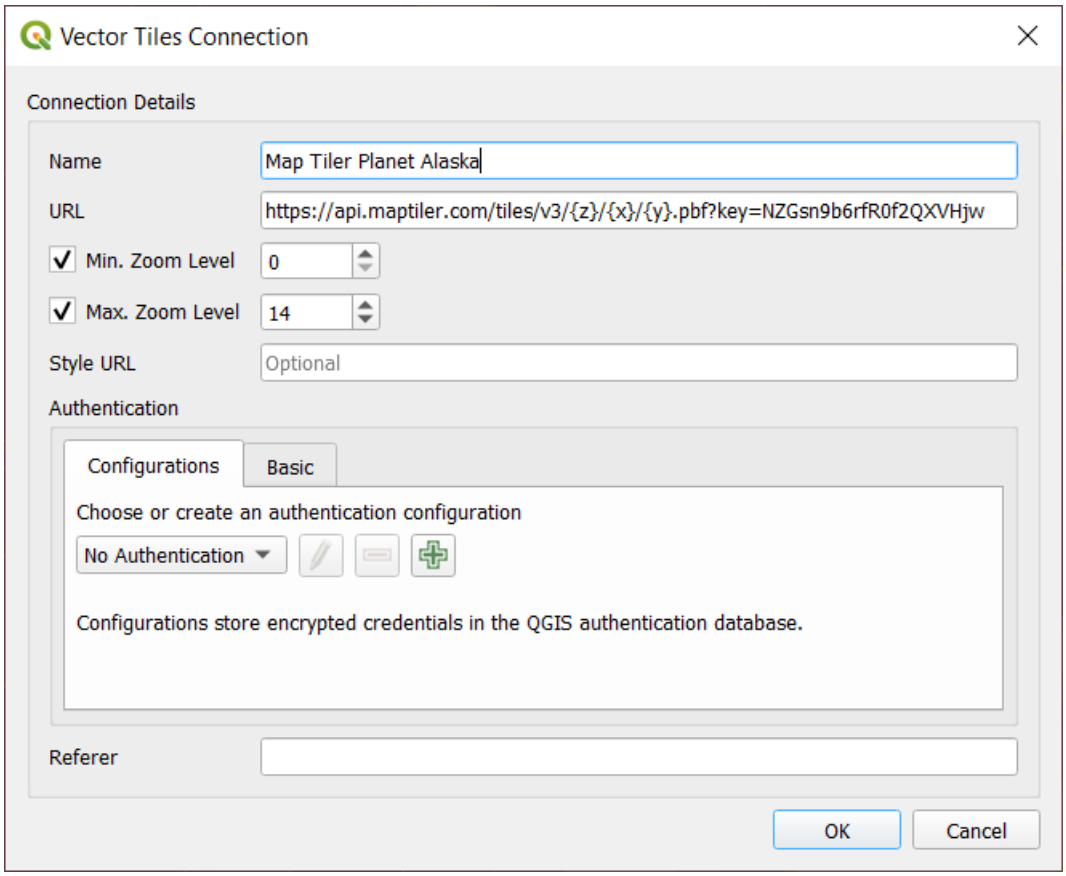


図 15.16: ベクタタイル - Maptiler Planet の設定

接続設定は、**データソースマネージャ** ダイアログ内の **ベクタタイル** エントリや、**ブラウザ** パネルの Vector Tiles のコンテキストメニューを通じて `.XML` ファイルに保存（ **接続を保存** ）することができます。同様に、接続をファイルから追加する（ **接続を読み込む** ）こともできます。

ベクタタイルサービスへの接続が設定できたら、以下の操作が可能です。

- ベクタタイルの接続設定の **編集**
- 接続の **削除**
- **ブラウザ** パネルでは、エントリを右クリックすることで、以下の操作もできます：
  - **レイヤをプロジェクトに追加する** ：ダブルクリックでもレイヤを追加できます
  - **レイヤのプロパティ**... を確認し、サービスによって提供されるメタデータやデータのプレビューにアクセスできます。レイヤがプロジェクトに読み込まれている場合には、より多くの設定が可能です。

XYZ タイルサービスを利用する

XYZ タイルサービスは、 **データソースマネージャ** ダイアログの  *XYZ* タブまたは **ブラウザ** パネルの *XYZ Tiles* エントリのコンテキストメニューから追加することができます。 **新規** （あるいは **新規接続** ）を押し、以下の情報を入力します：

- **名前**
- *URL*
- **認証** ：必要に応じて設定
- **最小ズームレベル** および **最大ズームレベル**
- **リファラー**
- **タイル解像度** ：可能な値は **不明（スケールされていない）** 、 **標準（***256x256 / 96DPI***）** および **高解像度（***512x512 / 192DPI***）** です
- **断面データの解釈**: WMTS/XYZ ラスタデータセットを、あらかじめ定義されたエンコーディングスキームに従って、シングルバンド float 型のラスタレイヤに変換します。サポートされているスキームは **デフォルト** (変換は行われません), *MapTiler Terrain RGB* および *Terrarium Terrain RGB* です。選択されたコンバータは、RGB ソースの値を各ピクセルの float 値に変換します。一度読み込まれると、レイヤはシングルバンドの浮動小数点ラスタレイヤとして表示され、QGIS の通常の:ref:**ラスタレンダラ** <*raster_rendering*> を使ってスタイルを設定することができます。

デフォルトで、QGIS はいくつかのすぐに使える XYZ Tiles サービスを提供しています：

-  *Mapzen Global Terrain* により、プロジェクトに使える世界的な DEM ソースに即座にアクセスできます。詳細とリソースは、https://registry.opendata.aws/terrain-tiles/
-  *OpenStreetMap* で世界の 2D マップにアクセスできます。|numref:*figure_xyz_tiles_openstreetmap* は OpenStreetMap XYZ Tile サービスの設定ダイアログを表示します。

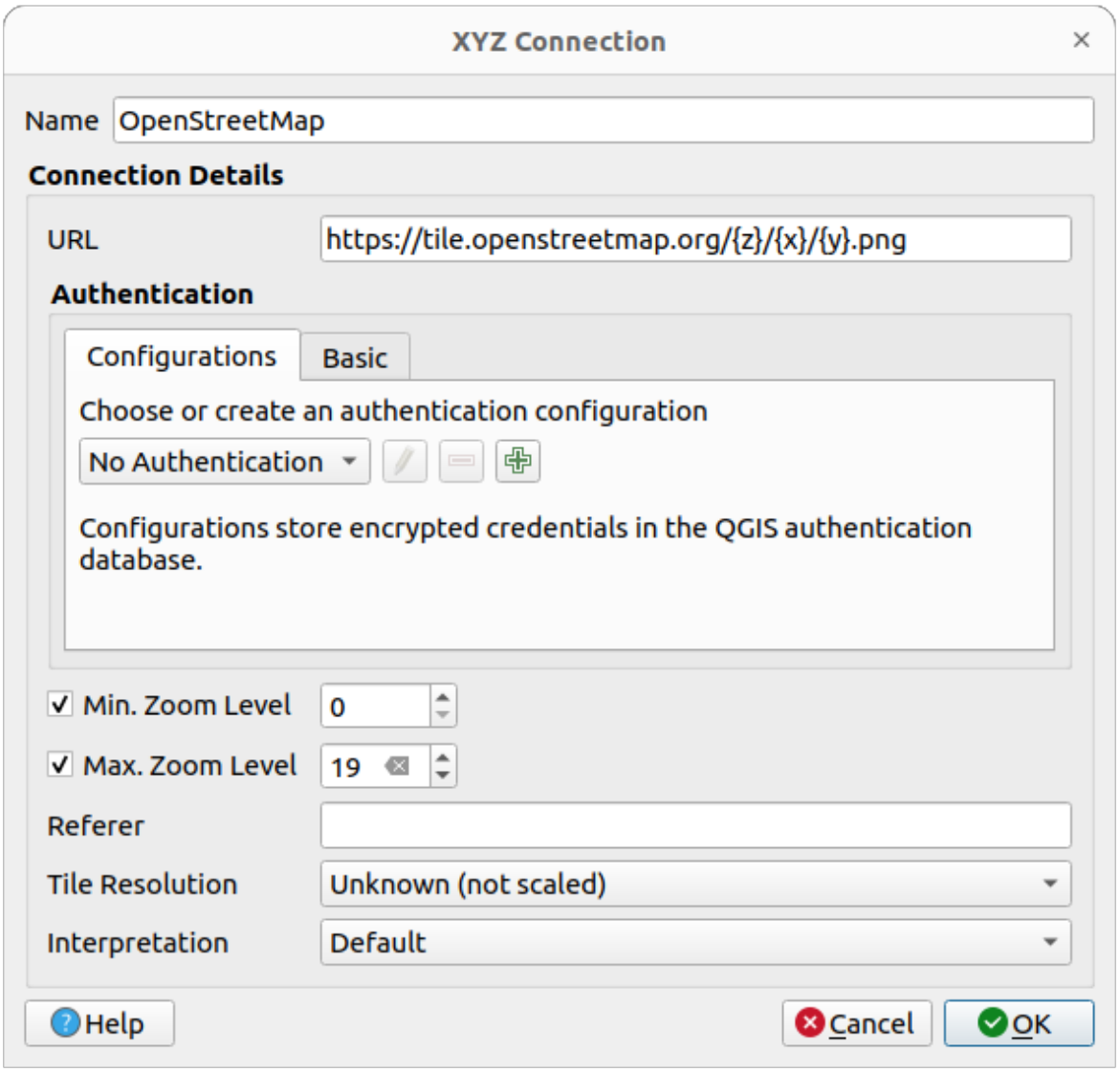


図 15.17: XYZ Tiles - OpenStreetMap の設定

接続設定は、**データソースマネージャ** ダイアログ内の **ベクタタイル** エントリや、**ブラウザ** パネルの XYZ Tiles のコンテキストメニューを通じて `.XML` ファイルに保存（**接続を保存**）することができます。同様に、接続をファイルから追加する（**接続を読み込む**）こともできます。

OpenStreetMap の XML ファイルは以下のようになっています：

```
<!DOCTYPE connections>
<qgsXYZTilesConnections version="1.0">
  <xyztiles url="https://tile.openstreetmap.org/{z}/{x}/{y}.png"
   zmin="0" zmax="19" tilePixelRatio="0" password="" name="OpenStreetMap"
   username="" authcfg="" referer=""/>
</qgsXYZTilesConnections>
```

XYZ タイルサービスへの接続が設定できたら、以下の操作が可能です。

- XYZ タイル接続設定の **編集**

- 接続の **削除**
- **ブラウザ** パネルでは、エントリを右クリックすることで、以下の操作もできます：
  - **レイヤエクスポート...** **ファイルへ** は、 タイルをラスタとして保存します
  - **レイヤをプロジェクトに追加する** ：ダブルクリックでもレイヤを追加できます
  - **レイヤのプロパティ...** を確認し、サービスによって提供されるメタデータやデータのプレビューにアクセスできます。レイヤがプロジェクトに読み込まれている場合には、より多くの設定が可能です。

XYZ タイルサービスの例：

- OpenStreetMap Monochrome: *URL*: `http://tiles.wmflabs.org/bw-mapnik/{z}/{x}/{y}.png`, **最小ズームレベル**: 0, **最大ズームレベル**: 19
- Google マップ: *URL*: `https://mt1.google.com/vt/lyrs=m&x={x}&y={y}&z={z}`, **最小ズームレベル**: 0, **最大ズームレベル**: 19
- Open Weather Map Temperature: *URL*: `http://tile.openweathermap.org/map/temp_new/{z}/{x}/{y}.png?appid={api_key}` **最小ズームレベル**: 0, **最大ズームレベル**: 19

ArcGIS REST Servers を利用する

ArcGIS REST Servers は、 **データソースマネージャ** ダイアログの  *ArcGIS REST Server* タブまたは **ブラウザ** パネルの *ArcGIS REST Servers* エントリのコンテキストメニューから追加することができます。 **新規**（あるいは **新規接続** ）を押し、以下の情報を入力します：

- **名前**
- *URL*
- *Community endpoint URL*
- *Content endpoint URL*
- 認証 ：必要に応じて設定
- **リファラー**

注釈: ArcGIS Feature Service 接続で、対応する Portal endpoint URL が設定されているものは、ブラウザパネル内でコンテンツグループごとに検索できます。

接続に Portal endpoints が設定されている場合、ブラウザパネルで接続を展開すると、通常表示されるサービスの完全なリストの代わりに “ Groups ” フォルダと “ Services ” フォルダが表示されます。グループフォルダを展開すると、ユーザーがメンバーとなっているすべてのコンテンツグループの一覧が表示され、それぞれのグループを展開すると、そのグループに所属するサービス項目が表示されます。

接続設定は、 **データソースマネージャ** ダイアログ内の *ArcGIS REST Server* エントリを通じて `.XML` ファイルに保存（ **接続を保存** ）することができます。同様に、接続をファイルから追加する（ **接続を読み込む** ）こともできます。

ArcGIS REST Server への接続が設定できたら、以下の操作が可能です。

- ArcGIS REST Server 接続設定の **編集**
- 接続の **削除**
- 接続の **再読み込み**
- 利用可能なレイヤへのフィルタ適用
-  **現在のビュー領域に重なる地物のみをリクエストする** オプションを用いて、利用可能なレイヤのリストから選択する
- **ブラウザ** パネルでは、接続エントリを右クリックすることで、以下の操作ができます：
  - **再読み込み**
  - **接続の編集**...
  - **接続を削除**...
  - **サービス情報を見る** ：デフォルトのウェブブラウザを開き、サービス情報を表示します
- レイヤのエントリを右クリックすると、以下の操作ができます：
  - **サービス情報を見る** ：デフォルトのウェブブラウザを開き、サービス情報を表示します
  - **レイヤエクスポート　ファイルへ…**
  - **レイヤをプロジェクトに追加する** ：ダブルクリックでもレイヤを追加できます
  - **レイヤのプロパティ**... を確認し、サービスによって提供されるメタデータやデータのプレビューにアクセスできます。レイヤがプロジェクトに読み込まれている場合には、より多くの設定が可能です。

15.2 レイヤを作成する

レイヤは下記も含め、さまざまな方法で作成することができます。

- ゼロから空のレイヤを作成する
- 既存のレイヤからレイヤを作成する
- クリップボードからレイヤを作成する
- 一つもしくは多数のレイヤに基づいた SQL ライクなクエリの結果としてレイヤを作成する（**仮想レイヤ** を参照）

また、QGIS は様々なフォーマットからインポートしたり、様々なフォーマットへエクスポートするためのツールも提供しています。

15.2.1 新しいベクタレイヤを作成する

QGIS では、さまざまな形式で新しいレイヤを作成できます。QGIS には GeoPackage、シェープファイル、SpatiaLite、GPX 形式、および一時スクラッチレイヤ（別名メモリレイヤ）を作成するためのツールがあります。 **新規** *GRASS* **レイヤ** の作成は GRASS プラグインでサポートされています。

新しい GeoPackage レイヤを作成する

新しい GeoPackage レイヤを作成するには、 **レイヤ** **レイヤの作成** メニューもしくは **データソースマネージャ** ツールバーの **新規** *GeoPackage* **レイヤ**... ボタンを押します。 図 15.18 に示すように、 **新規** *GeoPackage* **レイヤ** ダイアログが表示されます。

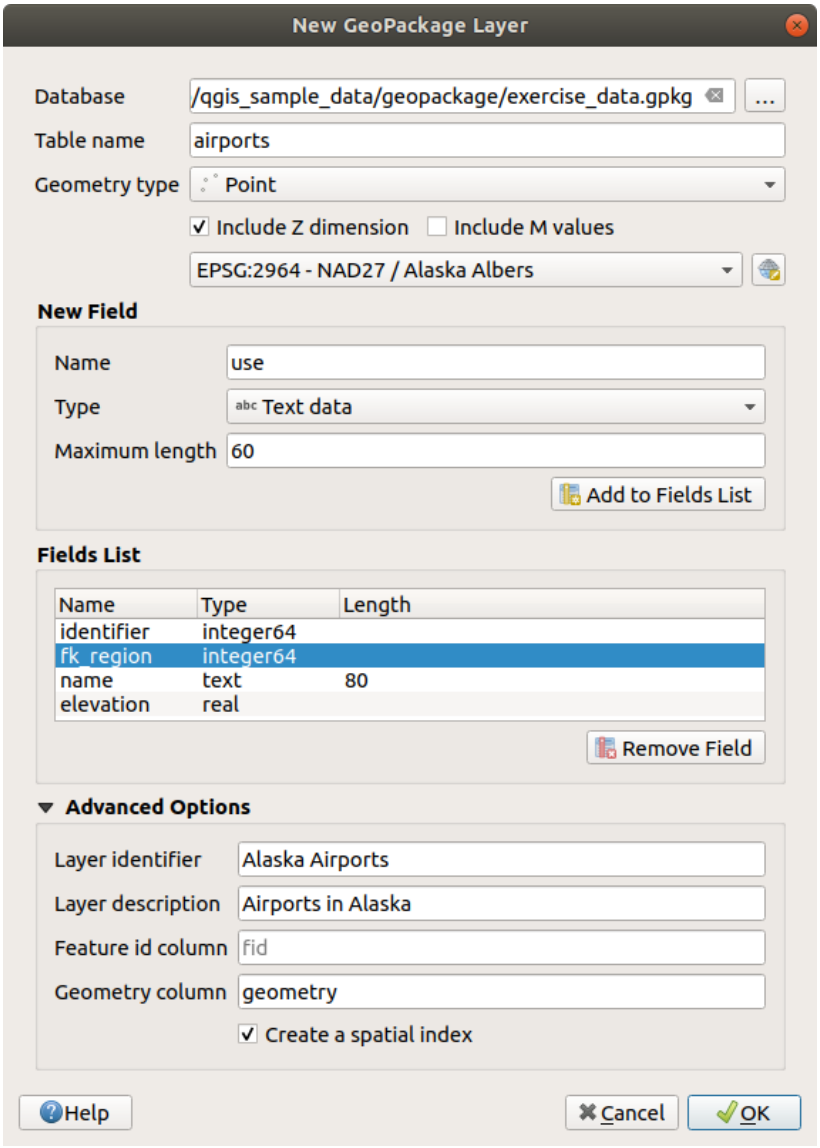


図 15.18: 新規 GeoPackage レイヤの作成ダイアログ

1. 最初のステップは、データベースファイルの場所を指定することです。 **データベース** フィールドの右にある ... ボタンを押して、既存の GeoPackage ファイルを選択するか、新しい GeoPackage ファイ

ルを作成します。QGIS は、指定した名前に適切な拡張子を自動的に追加します。

2. 新しいレイヤ / テーブルの名前を付けます（ **テーブル名** ）
3. **ジオメトリタイプ** を定義します。 ジオメトリなしのレイヤではない場合には、 *Z* **次元を含む** や *M* **値を含む** を指定できます。
4. ボタンを使用し、座標参照系を指定します

作成するレイヤにフィールドを追加するには、

1. フィールドの **名前** を入力します
2. データの **タイプ** を選択します。サポートされているタイプは、 **テキストデータ** 、 **整数値** （integer と integer64）、 **小数点付き数値** 、 **日付** と **日付時間** 、 **バイナリ（*BLOB*）** そして **ブール値** です。
3. 選択したデータタイプにもよりますが、値の **最大長さ** を入力します。
4. **フィールドリストに追加** ボタンをクリックします
5. 追加する必要のある各フィールドについて、上記の手順を繰り返します
6. 属性の設定に満足したら、 *OK* をクリックします。QGIS が新しいレイヤを凡例に追加し、 **既存レイヤのデジタイズ** で説明するように編集することができます。

デフォルトでは、GeoPackage レイヤを作成する際に QGIS は `fid` という名前の **地物 *ID* カラム** を作成します。このカラムはレイヤの主キーとして機能します。この名前は変更することが可能です。ジオメトリフィールドがある場合には `geometry` という名前になり、これに対して **空間インデックスを作成する** かを選択することができます。 これらのオプションは **高度なオプション** の下にあり、この他に **レイヤ識別子**（人間が読める短いレイヤ名）と **レイヤの説明** フィールドがあります。

GeoPackage レイヤのさらなる管理は、 *DB* **マネージャ** で行うことができます。

新しいシェープファイルレイヤを作成する

新しい ESRI シェープファイル形式のレイヤを作成するには、 **レイヤ** **レイヤの作成** メニューもしくは **データソースマネージャ** ツールバーの **新規シェープファイルレイヤ**... ボタンを押します。 図 15.19 に示すように、 **新規シェープファイルレイヤ** ダイアログが表示されます。

1. **ファイル名** の隣にある ... ボタンを使用し、ファイルパスとファイル名を指定します。QGIS は、指定した名前に適切な拡張子を自動的に追加します。
2. 次に、データの **ファイルエンコーディング** を指定します
3. レイヤの **ジオメトリタイプ** を選択します：ジオメトリなし（ `.DBF` 形式ファイルのみ）、ポイント、マルチポイント、ライン、ポリゴン
4. ジオメトリが追加次元を持っているかを指定します： **なし** 、 *Z* **値（＋** *M* **値）** 、 *M* **値**
5. ボタンを使用し、座標参照系を指定します

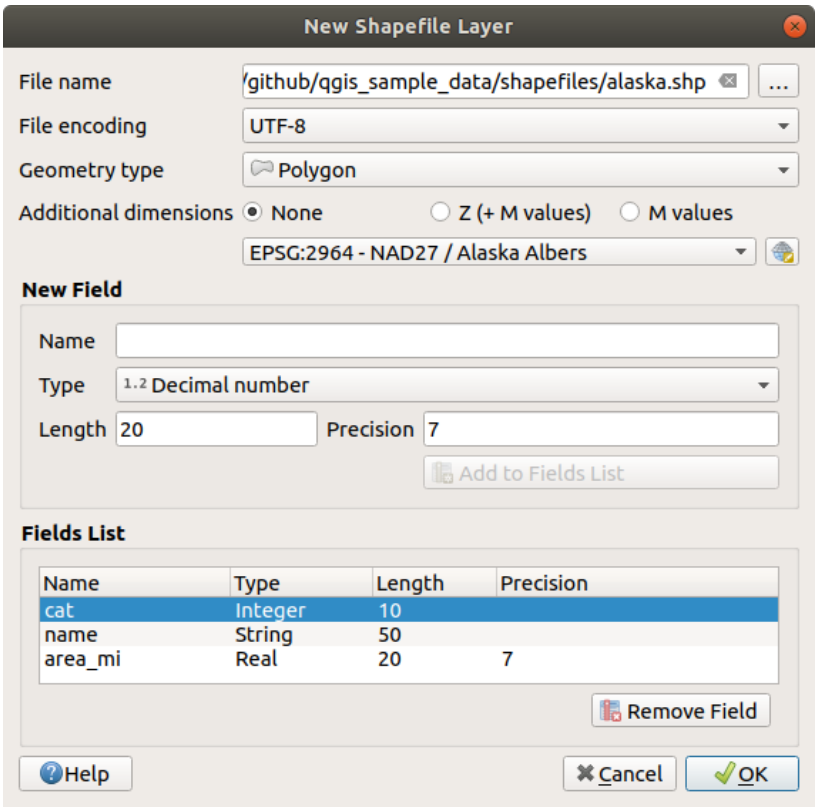


図 15.19: 新規シェープファイルレイヤ作成ダイアログ

作成するレイヤにフィールドを追加するには、

1. フィールドの **名前** を入力します
2. データの **タイプ** を選択します。 **小数点付き数値** 、 **整数値** 、 **テキストデータ** 、 **日付** 属性のみをサポートしています。
3. 選択したデータタイプに応じて、 **長さ** と **精度** を入力します。
4. **フィールドリストに追加** ボタンをクリックします
5. 追加する必要のある各フィールドについて、上記の手順を繰り返します
6. 属性の設定に満足したら、 *OK* をクリックします。QGIS が新しいレイヤを凡例に追加し、 **既存レイヤのデジタイズ** で説明するように編集することができます。

デフォルトでは、最初に整数値タイプの `id` カラムが追加されますが、これは削除することができます。

新しい SpatiaLite レイヤを作成する

新しい SpatiaLite レイヤを作成するには、 **レイヤ** **レイヤの作成** メニューもしくは **データソースマネージャ** ツールバーの **新規** *SpatiaLite* **レイヤ**... ボタンを押します。 図 15.20 に示すように、 **新しい** *SpatiaLite* **レイヤ** ダイアログが表示されます。

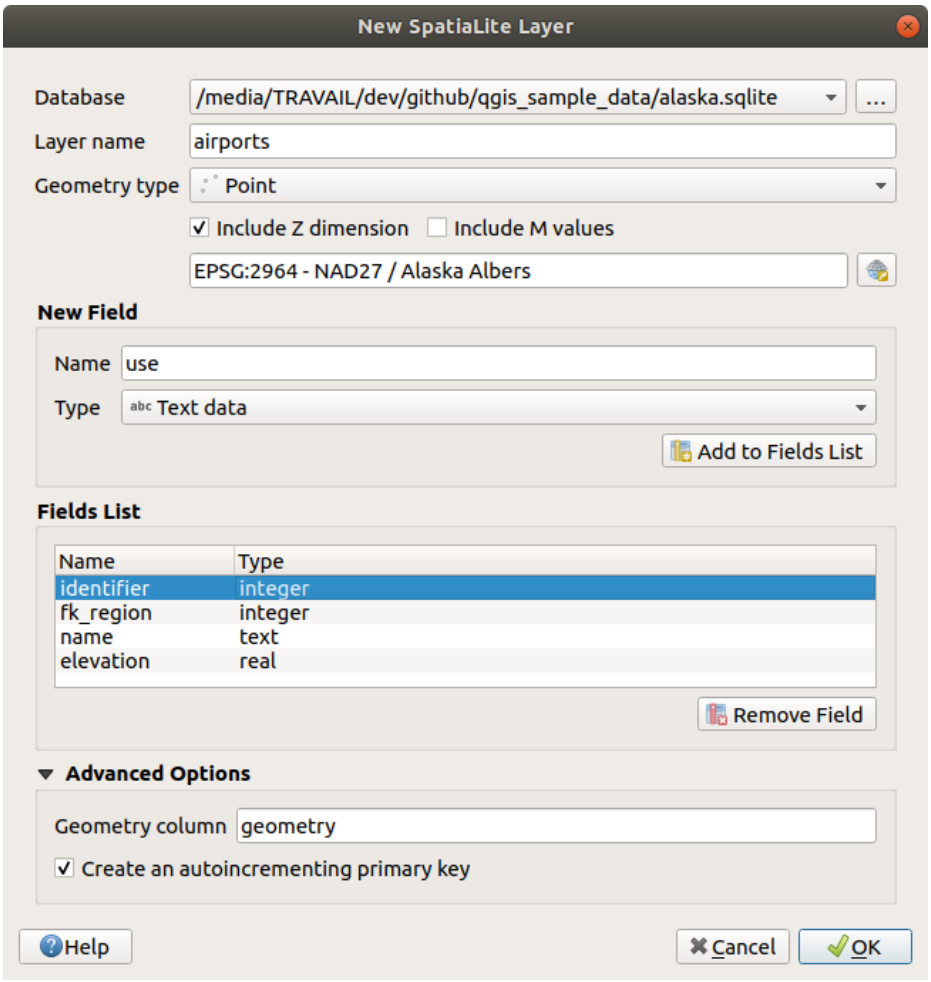


図 15.20: 新規 SpatiaLite レイヤの作成ダイアログ

1. 最初のステップは、データベースファイルの場所を指定することです。 **データベース** フィールドの右にある ... ボタンを押して、既存の SpatiaLite ファイルを選択するか、新しい SpatiaLite ファイルを作成します。QGIS は、指定した名前に適切な拡張子を自動的に追加します。
2. 新しいレイヤの名前（ **レイヤ名** ） を指定します
3. **ジオメトリタイプ** を定義します。 ジオメトリなしのレイヤではない場合には、 *Z* **次元を含む** や *M* **値を含む** を指定できます。
4. ボタンを使用し、座標参照系を指定します

作成するレイヤにフィールドを追加するには、

1. フィールドの **名前** を入力します
2. **データ型** を選択します。サポートしている型は、 **テキストデータ** 、 **整数値** 、 **小数点付き数値** 、 **日**

付 および **日付時刻** です。

3. **フィールドリストに追加** ボタンをクリックします
4. 追加する必要のある各フィールドについて、上記の手順を繰り返します
5. 属性の設定に満足したら、*OK* をクリックします。QGIS が新しいレイヤを凡例に追加し、**既存レイヤのデジタイズ** で説明するように編集することができます。

必要に応じて、**詳細オプション** セクション下にある **自動インクリメントするプライマリキーを作成する** を選択することができます。また、**ジオメトリカラム** の名前（デフォルトは `geometry` ）を変えることもできます。

SpatiaLite レイヤのさらなる管理は、*DB* **マネージャ** で行うことができます。

新しいメッシュレイヤを作成する

新しいメッシュレイヤを作成するには、**レイヤ　レイヤを作成　** メニューもしくは **データソースマネージャ** ツールバーの **新規メッシュレイヤ...** ボタンを押します。図 15.21 に示すように、**新規メッシュレイヤ** ダイアログが表示されます。

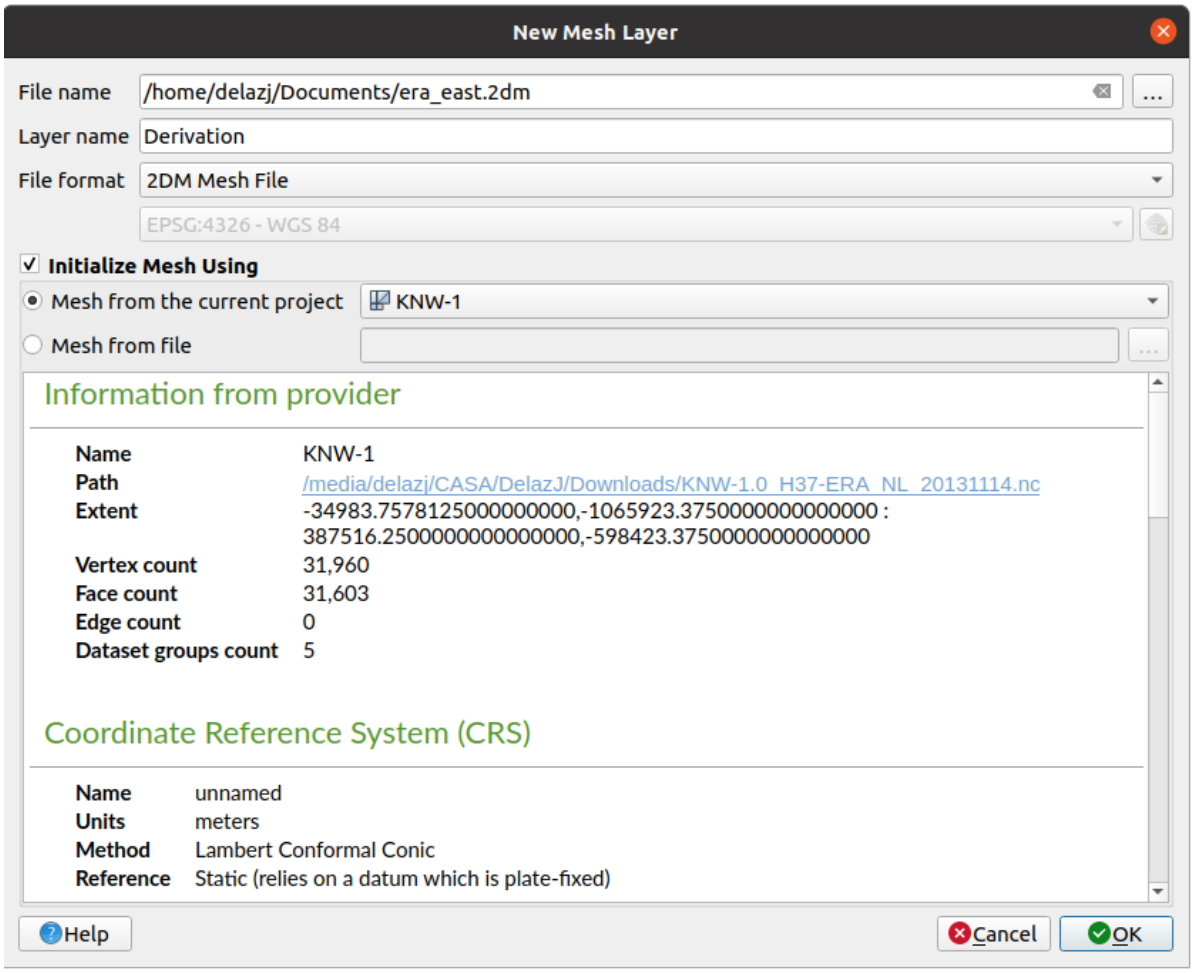


図 15.21: 新規メッシュレイヤの作成ダイアログ

1. 最初のステップは、メッシュファイルの場所を指定することです。 **ファイル名** フィールドの右にある ... ボタンを押して、既存のメッシュファイルを選択するか、新しいメッシュファイルを作成します。
2. 名前（ **レイヤ名** ）、つまりは **レイヤ** パネルに表示されるレイヤの名前を指定します。
3. **ファイル形式** を選択します。現在のところサポートしているメッシュファイル形式は、 `2DM Mesh File (*.2dm)` 、 `Selafin File (*.slf)` および `UGRID (*.nc)` です。
4. データセットに割り当てる **座標参照系** を指定します。
5. 上記のステップによって空のレイヤが生成され、その後、頂点をデジタイズしたりデータセットグループを追加できます。一方で、既存のメッシュレイヤを使用して初期化する、つまり、新しいレイヤに既存のレイヤの頂点や面を取り込むことも可能です。これを行うためには、次のようにします：
   1. **メッシュ初期化の方法** にチェックを入れます
   2. **現在のプロジェクトのメッシュ** または **ファイルから** を選択します。確認のために、選択したメッシュファイルの情報が表示されます。

   新規メッシュレイヤにコピーされるのは、メッシュレイヤのフレームのみであることに注意してください。データセットはコピーされません。

新しい GPX レイヤを作成する

新しい GPX ファイルを作成するには：

1. **レイヤ** メニューから **レイヤを作成** **新規** *GPX* **レイヤ**... を選択します。
2. ダイアログで新しいファイルを保存する場所を選択し、ファイル名を指定して **保存** ボタンを押します。
3. **レイヤパネル** に新しく 3 つのレイヤが追加されます。
   - 位置をデジタイズするポイントレイヤ（ `waypoints` ）。 name、elevation、comment、description、source、url、url name を保存するフィールドを持ちます。
   - 計画されたルートを構成する、一連の位置をデジタイズするラインレイヤ（ `routes` ）。name、symbol、number、comment、description、source、url、url name を保存するフィールドを持ちます。
   - 受信機の時間的な動きをたどるラインレイヤ（ `tracks` ）。name、symbol、number、comment、description、source、url、url name を保存するフィールドを持ちます。
4. これらのレイヤのいずれも、 **既存レイヤのデジタイズ** のセクションで説明されている方法で編集できます。

新しい一時スクラッチレイヤを作成する

一時スクラッチレイヤはインメモリレイヤであり、ディスク上には保存されておらず、QGIS を終了する時に破棄されます。これは一時的に必要な地物を保持したり、ジオプロセシング操作中の中間レイヤとして使用する場合に便利です。

新しい一時スクラッチレイヤを作成するには、 **レイヤ** **レイヤの作成** メニューもしくは **データソースマネージャ** ツールバーの **新しい一時スクラッチレイヤ...** エントリを選択します。 図 15.22 に示すように、 **新規一時スクラッチレイヤ** ダイアログが表示されます。その後、次の手順で操作します：

1. **レイヤ名** を指定します
2. **ジオメトリタイプ** を選択します。下記タイプのレイヤを作成できます：
   - ジオメトリなし タイプのレイヤは、シンプルなテーブルです
   - ポイント または マルチポイント レイヤ
   - ラインストリング / 複合曲線 または マルチラインストリング / マルチカーブ レイヤ
   - ポリゴン / カーブポリゴン または マルチポリゴン / マルチサーフェス レイヤ
3. ジオメトリタイプについて、データセットの次元を定義します：*Z* **次元を含む** かつ/または *M* **値を含める** かどうかをチェックボックスで設定します。
4. ボタンを使用し、座標参照系を指定します
5. レイヤにフィールドを追加します。他の多くの形式とは異なり、一時スクラッチレイヤはフィールドなしでも作成できることに注意してください。従ってこのステップは任意です。
   1. フィールドの **名前** を入力します
   2. データの **型** を選択します：**テキスト**、**整数値**、**小数点付き数値**（*decimal*）、**ブール値**（*boolean*）、**日付**（*Date*）、**時刻**（*Time*）、**日付時刻**（*Datetime*）そして **バイナリ**（*BLOB*）型をサポートしています。
   3. 選択したデータ型に応じて、 **長さ** や **精度** を入力します。
   4. **フィールドリストに追加** ボタンをクリックします
   5. 追加する必要のある各フィールドについて、上記の手順を繰り返します
6. 設定に満足したら、 *OK* をクリックします。QGIS は新しいレイヤを **レイヤ** パネルに追加し、 **既存レイヤのデジタイズ** のセクションで説明するように編集することができます。

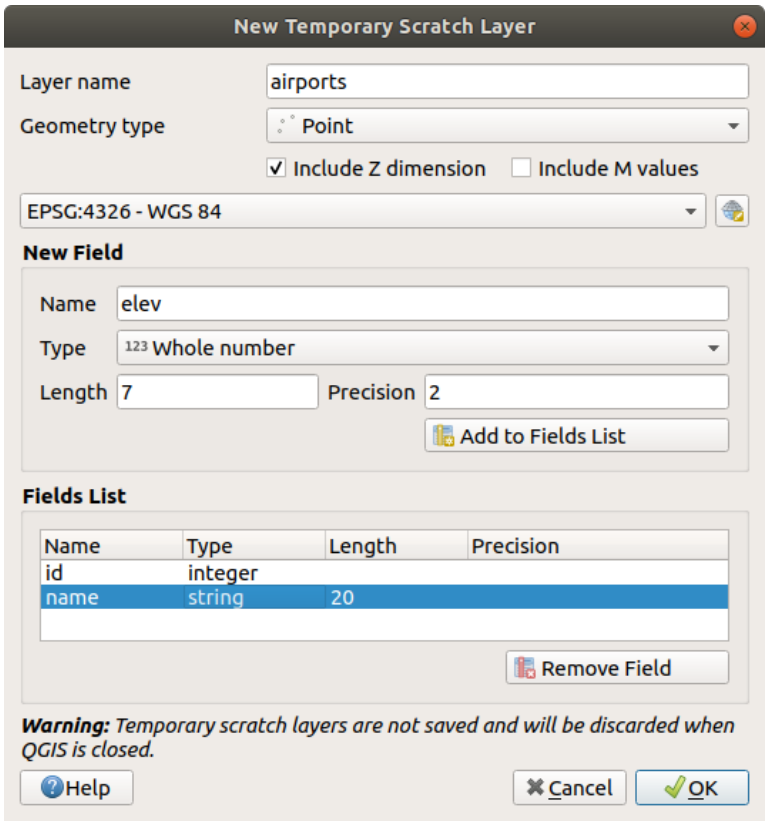


図 15.22: 新しい一時スクラッチレイヤの作成ダイアログ

また、フィールド入力済みの一時レイヤを作成する方法には、例えば、クリップボードを使用して（**クリップボードから新しいレイヤを作成する**）や、**プロセシングアルゴリズム** の出力結果などもあります。

Tip: メモリレイヤをディスク上に永続的に保存する

プロジェクトを閉じたときに一時スクラッチレイヤのデータが失われないようにするために、これらのレイヤを次の方法で QGIS がサポートする任意のベクタ形式に保存することができます：

- レイヤの横にある 表示アイコンをクリックする
- レイヤのコンテキストメニューの **保存...** エントリを選択する
- レイヤのコンテキストメニューの **エクスポート** エントリ、または **レイヤ** **名前を付けて保存...** メニューを使用する

これらのコマンドはいずれも、**既存のレイヤから新しいレイヤを作成する** セクションで説明する **ベクタレイヤを名前を付けて保存...** ダイアログを開きます。保存されたファイルは **レイヤ** パネルの一時スクラッチレイヤを置き換えます。

15.2.2 既存のレイヤから新しいレイヤを作成する

ラスタレイヤ、ベクタレイヤともに、異なる形式で保存したり、異なる座標参照系（CRS）に再投影することができます。これには、**レイヤ　名前をつけて保存**... メニューを使用するか、**レイヤパネル** のレイヤを右クリックして次のいずれかを選択します：

- ラスタレイヤの場合は **エクスポート　名前を付けて保存**...
- ベクタレイヤの場合は **エクスポート　地物の保存**... または **エクスポート　選択地物の保存**...
- レイヤツリーからレイヤを **ブラウザパネル** の PostGIS エントリへとドラッグ＆ドロップします。**ブラウザパネル** で PostGIS への接続が必要であることに注意してください。

共通のパラメータ

名前を付けて保存... ダイアログには、レイヤを保存する際の動作を変更するパラメータがいくつかあります。ラスタとベクタで共通のパラメータは次のとおりです：

- **ファイル名** ：ディスク上のファイルの場所です。これは出力レイヤ、またはレイヤを格納するコンテナ（例えば GeoPackage や SpatiaLite、Open Document Spreadsheets といったデータベースライクな形式）を参照できます。
- *CRS* ：変更して、データを再投影することができます。
- **領域** ： **範囲セレクタ** ウィジェットを使用して、エクスポートされる入力の範囲を制限します
- **保存されたファイルを地図に追加する** ：新しいレイヤをマップキャンバスに追加します。

一方、ラスタ形式とベクタ形式で固有のパラメータもあります：

ラスタに固有のパラメータ

エクスポートの形式によっては、これらのオプションが利用できない場合があります：

- **出力モード** （ **生データ** または **画像** ）
- **形式** ：GeoTiff、GeoPackage、MBTiles、Geospatial PDF、SAGA GIS Binary Grid、Intergraph Raster、ESRI .hdr Labelled など、GDAL が書き出せる任意のラスタ形式にエクスポートします。
- **解像度**
- **作成オプション** ：ファイルを生成する際に、出力フォーマットに関連した **既定の作成プロファイル** を使用するか、各パラメータを設定することで高度なオプション（ファイル圧縮、ブロックサイズ、測色など）を使用します。
- **ピラミッド** 作成
- *VRT* **タイル** ： *VRT* **作成** にチェックを入れた場合
- *nodata* **値**

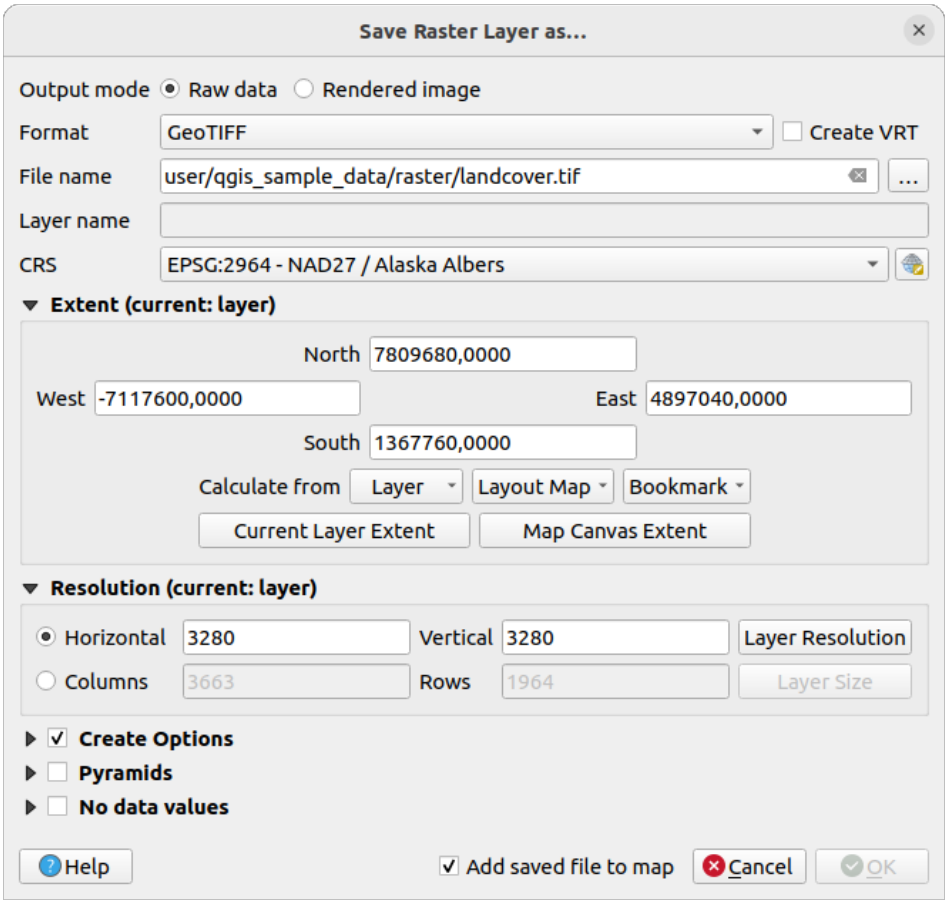


図 15.23: 新しいラスタレイヤとして保存する

ベクタに固有のパラメータ

エクスポートの形式によりますが、これらのオプションが利用できます：

- **形式**：GeoPackage、GML、ESRI Shapefile、AutoCAD DXF、ESRI FileGDB、Mapinfo TAB や Mapinfo MIF、SpatiaLite、CSV、KML、ODS など、GDAL が書き出せる任意のベクタ形式にエクスポートします。
- **レイヤ名** ： **ファイル名** がコンテナライクな形式を参照している場合には、このエントリは出力レイヤの名前を表します。
- **文字コード**
- **選択地物のみ保存**
- **エクスポートするフィールドとエクスポートオプションを選択**: カスタム名と **フォームウィジェット** 設定を持つフィールドをエクスポートする手段を提供します：
  - **名前** カラムの下の行をチェックして、出力レイヤに残すフィールドを選択するか、 **すべて選択** または **すべての選択を解除** ボタンを押します
  - **エクスポート名にエイリアスを使う** チェックボックスを切り替えると、 **エクスポート名** カラムに対応するフィールドのエイリアスが入力されるか、元のフィールド名にリセットされます。

セルをダブルクリックすると名前の編集もできます。

- 属性フォームカスタムウィジェットが使用されているかどうかによって、**選択したすべての** *Raw* **フィールド値を表示値で置き換える** ことができます。例えば、`value map` ウィジェットがフィールドに適用されている場合、出力レイヤは元の値の代わりに説明の値を含みます。置換はフィールドごとに **表示の値で置き換える** カラムで行うこともできます。

- **レイヤメタデータを保持** ：ソースレイヤに存在する、レイヤの任意の **メタデータ** がコピーして保存されます
  - 出力が GeoPackage 形式の場合は、新規に作成されるレイヤ内に保存されます
  - その他の形式の場合は、出力レイヤとともに `.qmd` ファイルとして保存されます。複数のデータセットをサポートするファイルベースの形式（例：SpatiaLite、DXF 等）の場合には、意図しない動作をすることがあることに注意してください。

- **シンボロジのエクスポート** ：主に DXF のエクスポートに使われますが、DXF、KML、tab ファイル形式など、OGR Feature Styles（下記注を参照）を扱うことのできるすべてのファイル形式で使用できます。
  - **シンボロジなし** ：データ読み込むアプリケーションのデフォルトのスタイル
  - **地物シンボロジ** ：OGR Feature Styles によるスタイルの保存（下記を参照）
  - **シンボルレイヤシンボロジ** ：OGR Feature Styles で保存（下記の注を参照）しますが、レイヤがマルチシンボロジのシンボルを使用している場合、同じジオメトリを複数回エクスポートします
  - **縮尺** の値は、現在のキャンバス縮尺値を設定することができます。

注釈: *OGR Feature Styles* は、スタイルを隠し属性としてデータ内に直接保存する方法です。この種の情報を扱うことができるのは一部のファイル形式だけです。KML、DXF、TAB ファイル形式がそのような形式です。さらなる詳細については、 OGR Feature Styles specification ドキュメントを参照してください。

- **ジオメトリ** ：出力レイヤのジオメトリ機能を設定できます
  - **ジオメトリタイプ** ： **自動** に設定されている場合には地物のオリジナルのジオメトリを保持し、そうでない場合には削除または任意のジオメトリタイプで上書きします。属性テーブルに空のジオメトリカラムを追加し、空間レイヤのジオメトリカラムを削除することもできます。
  - **マルチタイプにする** ：レイヤにマルチジオメトリ地物を強制的に作成します。
  - *Z* **次元を含める** ：ジオメトリに Z 次元を含めます。

Tip: レイヤのジオメトリタイプを上書きすることで、ジオメトリを持たないテーブル（例えば `.csv` ファイル）を任意のタイプのジオメトリ（ポイント、ライン、ポリゴン）を持ったシェープファイルに保存して、 部分の追加 ツールを使用してジオメトリを手動で行に追加できるようになります。

- **データソースオプション** 、 **レイヤオプション** 、 **カスタムオプション** では、出力形式に応じて高度なパラメータ設定を行えます。一部は **データ形式とフィールドを探究する** で説明されていますが、完全な詳細は GDAL のドライバドキュメントを参照してください。ファイル形式ごとににそれぞれ独

自のカスタムパラメータを持っています。例えば、 GeoJSON 形式については、 GDAL GeoJSON のドキュメントを参照してください。

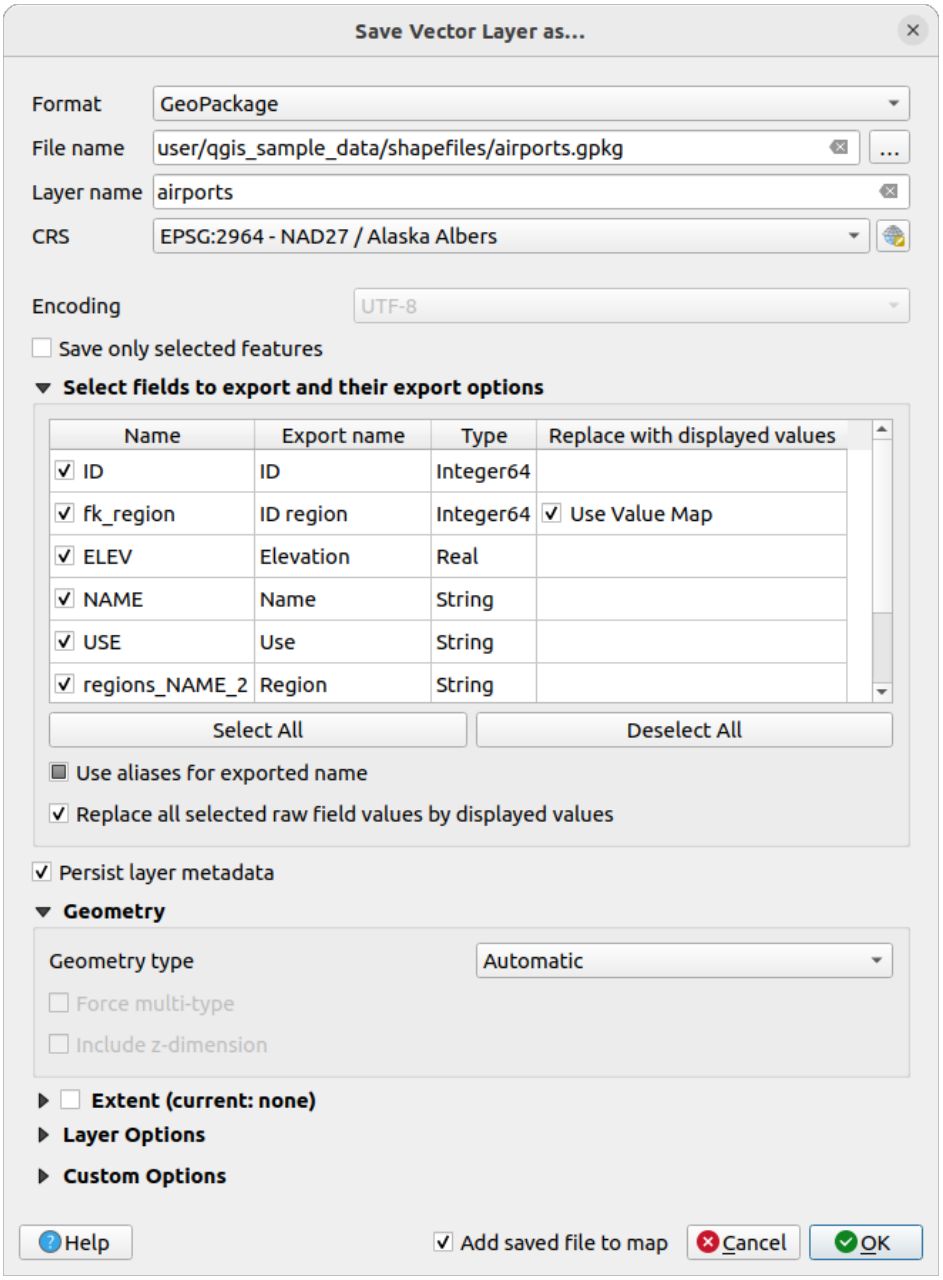


図 15.24: 新しいベクタレイヤとして保存する

ベクタレイヤを既存のファイルに保存する場合、出力形式（GeoPackage、SpatiaLite、FileGDB 等）の機能に応じて、ユーザーは下記操作を行うかどうかを決めることができます：

- ファイル全体を上書きする
- ターゲットレイヤだけを上書きする（レイヤ名は設定可能です）
- 既存のターゲットレイヤに地物を追加する
- 地物を追加し、新しいフィールドがあれば追加する

ESRI シェープファイル、MapInfo .TAB 等の形式は、地物の追加も可能です。

15.2.3 新しい DXF ファイルを作成する

単一のレイヤを `*.DXF` を含む別の形式にエクスポートするオプションを提供する **名前を付けて保存**... ダイアログの他に、QGIS は複数のレイヤを単一の DXF レイヤとしてエクスポートするための別のツールを提供しています。これは **プロジェクト** ▸ **インポートとエクスポート** ▸ **プロジェクトを *DXF* にエクスポート**... メニューからアクセスできます。

***DXF* エクスポート** ダイアログで以下の設定を行います：

1. 保存先のファイルを指定します。
2. 該当する場合は、シンボロジーモードとシンボルの縮尺を選択します（ *OGR Feature Styles* の注釈参照）。
3. データの **文字コード** を選択します。
4. 適用する *CRS* を選択します。選択したレイヤは指定した CRS に再投影されます。
5. DXF ファイルに含めるレイヤを、テーブルウィジェットでチェックを付けるか、既存の **地図のテーマ** からの自動抽出で選択します。 **すべて選択** ボタンと **すべての選択を解除** ボタンは、エクスポートするデータを素早く設定するのに役立ちます。

   各レイヤについて、DXF 出力時に全ての地物を単一の DXF レイヤにエクスポートするか、もしくはフィールドの値に依存してして地物をレイヤに分割するかを選択できます。

下記オプションを選択することもできます：

- レイヤ名自体の代わりに、 **設定されている場合はレイヤタイトルを名前として使う**
- **現在の地図領域に交差する地物をエクスポートする**
- *2D* **出力を強制する（例えばポリライン幅をサポートするため）**
- **ラベルを *MTEXT* 要素としてエクスポートする** ：チェックなしでは TEXT 要素です。

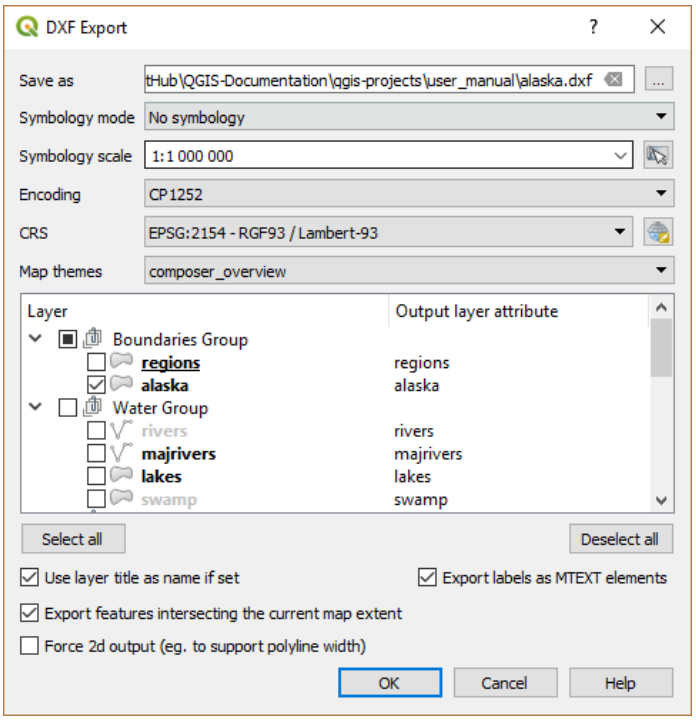


図 15.25: プロジェクトを DXF にエクスポートダイアログ

15.2.4 クリップボードから新しいレイヤを作成する

クリップボードにある地物は新しいレイヤに貼り付けできます。これを行うには、いくつかの地物を選択してクリップボードにコピーし、**編集** **新規レイヤへの地物貼り付け** で以下のどちらかを選択することで、新しいレイヤに地物を貼り付けます。

- **新規ベクタレイヤ...** ： **ベクタレイヤを名前を付けて保存...** ダイアログが現れます（パラメータについては **既存のレイヤから新しいレイヤを作成する** を参照）
- **一時スクラッチレイヤ...** ：レイヤ名を指定する必要があります

選択した地物とその属性を持った新しいレイヤが作成され（そしてマップキャンバスに追加され）ます。

注釈: クリップボードからのレイヤ作成は、well-known text（WKT）を使用してジオメトリが定義されていれば、QGIS 内で選択してコピーした地物でも、ほかのアプリケーションの地物でも可能です。

15.2.5 仮想レイヤを作成する

仮想レイヤは特別な種類のベクタレイヤです。これは、QGIS が開くことができる任意の数の他のベクタレイヤに関する SQL クエリの結果としてレイヤを定義します。仮想レイヤは、それ自体はデータを保持しておらず、ビューと見なすことができます。

仮想レイヤを作成するためには、下記の方法で仮想レイヤの作成ダイアログを開きます：

- **レイヤ** **レイヤの追加** メニュー内の **仮想レイヤの追加 / 編集** エントリを選択する
- **データソースマネージャ** ダイアログ内で **仮想レイヤ** タブを有効にする
- *DB* **マネージャ** ダイアログのツリーを使用する

ダイアログでは、 **レイヤ名** と SQL **クエリ** を指定します。クエリでは、読み込まれたベクタレイヤの名前（または id）をテーブルとして、フィールド名をカラムとして使用することができます。

例えば、 `airports` という名前のレイヤがある場合、 `public_airports` という名前の新しい仮想レイヤを次のような SQL クエリで作成できます。

```
SELECT *
FROM airports
WHERE USE = "Civilian/Public"
```

この SQL クエリは `airports` レイヤの元のプロバイダが直接には SQL クエリをサポートしていなくとも、プロバイダに関係なく実行されます。

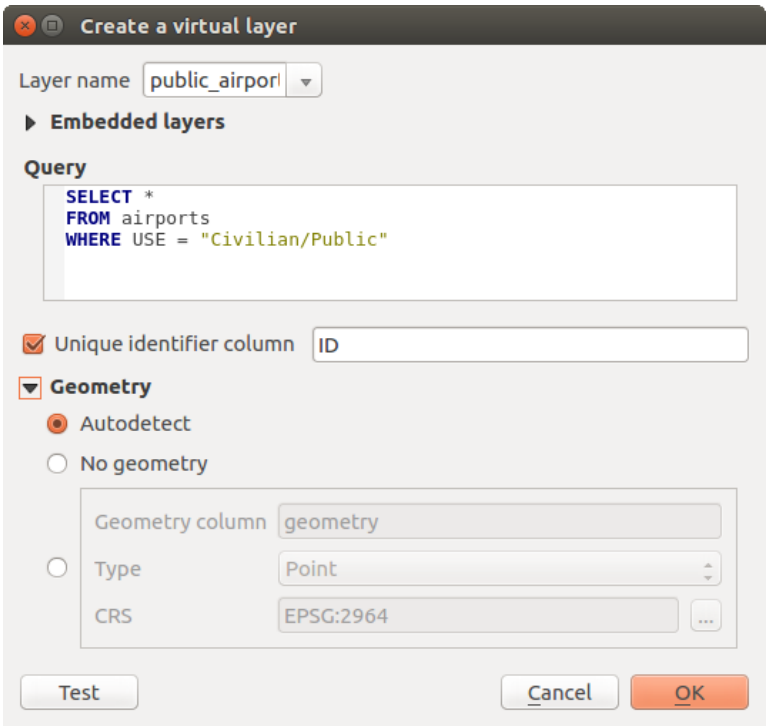


図 15.26: 仮想レイヤの作成ダイアログ

結合（JOIN）や、複雑なクエリを使用して作成することもできます。例えば、 airports と country の情報を結合するには、次のようにします：

```
SELECT airports.*, country.population
FROM airports
JOIN country
ON airports.country = country.name
```

注釈: *DB* **マネージャプラグイン** の SQL ウィンドウを使用して仮想レイヤを作成することもできます。

クエリで使用するためのレイヤの埋め込み

マップキャンバスで利用できるベクタレイヤの他に、**埋め込みレイヤ** リストにレイヤを追加することができます。埋め込みレイヤは、マップキャンバスやレイヤパネルに表示することなくクエリ内で使用することができます。

レイヤを埋め込むには、**追加** をクリックし **ローカル名** 、 **プロバイダ** 、 **エンコーディング** および **ソース** へのパスを指定します。

インポート ボタンを押すと、マップキャンバスのレイヤを埋め込みレイヤリストに追加できます。そのレイヤは既存のクエリを壊すことなくレイヤパネルから削除できます。

サポートされているクエリ言語

クエリ操作の基本エンジンは SQLite と SpatiaLite を使用しています。

これはすなわち、ローカルインストールの SQLite が理解できる SQL はすべて使用できることを意味します。

SQLite の関数や SpatiaLite の空間関数も仮想レイヤのクエリで使用できます。例えば、属性のみのレイヤからポイントレイヤを作成するには、次のようなクエリを使用します：

```
SELECT id, MakePoint(x, y, 4326) as geometry
FROM coordinates
```

QGIS **式の関数** も仮想レイヤのクエリ内で使用できます。

レイヤのジオメトリ列を参照するため、名前 `geometry` を使用します。

純粋な SQL クエリとは違い、仮想レイヤのクエリのフィールドには全て名前を付けなければなりません。計算や関数呼び出しの結果としてカラムに名前を付ける場合には、`as` キーワードを使用することを忘れないでください。

パフォーマンスの問題

デフォルトのパラメータを使用すると、仮想レイヤのエンジンは、クエリのさまざまなカラムの型を検出するために最善を尽くそうとします。もしある場合には、ジオメトリカラムも含め型検出を行います。

型検出は、可能な場合にはクエリをイントロスペクトするか、最後の手段としてクエリの最初の行（LIMIT 1）をフェッチすることで行われます。レイヤを作成するためだけに結果の最初の行をフェッチするのは、パフォーマンス上好ましくないかもしれません。

作成ダイアログのパラメータには以下のものがあります：

- **ユニークな識別子カラム** ：QGIS が行の識別子として使用可能なユニークな整数値を表すクエリのフィールドを指定します。デフォルトでは、自動的インクリメントの整数値が使用されます。ユニークな識別子カラムを定義すると、ID による行の選択を高速化します。
- **ジオメトリなし** ：仮想レイヤのジオメトリフィールドを強制的に無視します。結果として得られるレイヤは属性のみのレイヤとなります。
- **ジオメトリカラム** ： ジオメトリカラムの名前を指定します。
- **タイプ** ：ジオメトリの種類を指定します。
- *CRS* ：仮想レイヤの座標参照系を指定します。

特別なコメント

仮想レイヤエンジンは、クエリの各列の型を決定しようとします。それが失敗した場合、列の型を決定するためにクエリの最初の行をフェッチします。

いくつかの特別なコメントを使用することで、特定の列の型をクエリ内で直接指定できます。

構文は、 `/*:type*/` のようになります。これはカラム名の直後に記述しなければなりません。 `type` は、整数の場合は `int` 、浮動小数点数の場合は `real` 、テキストの場合は `text` です。

例：

```
SELECT id+1 as nid /*:int*/
FROM table
```

ジオメトリカラムの型と座標参照系は、構文 `/*:gtype:srid*/` 形式の特別なコメントで設定できます。ここで、 `gtype` はジオメトリタイプ（ `point` 、 `linestring` 、 `polygon` 、 `multipoint` 、 `multilinestring` 、 `multipolygon` ）、 `srid` は座標参照系の EPSG コードを表す整数です。

インデックスの使用

仮想レイヤを介してレイヤをリクエストする場合、ソースレイヤのインデックスは以下の方法で使用されます：

- レイヤの主キーカラムに = 述語が使用されている場合、元のデータプロバイダには、特定の ID（FilterFid）が要求されます。
- 他の述語（ > 、 <= 、 != など）や、主キーのないカラムについては、元のベクタデータプロバイダへのリクエストには、式から構築されたリクエストが使用されます。これは、データベースプロバイダにインデックスがある場合には、インデックスが使用される可能性があることを意味します。

リクエスト内の空間的述語を処理し、空間インデックスの使用をトリガーするための特定の構文が存在します： `_search_frame_` という名前の隠しカラムが各仮想レイヤーに対して存在します。このカラムは、バウンディングボックスに等しいかどうかを比較できます。例：

```
SELECT *
FROM vtab
WHERE _search_frame_=BuildMbr(-2.10,49.38,-1.3,49.99,4326)
```

`ST_Intersects` のような空間二項述語は、この空間インデックス構文と組み合わせて使用すると、大幅に高速化されます。

15.3 データ形式とフィールドを探究する

15.3.1 ラスタデータ

GIS ラスタデータは、地表面上、大気中、地下の特徴や現象を表す離散的なセルの行列です。ラスタグリッドの各セルは同じサイズで、セルは通常、長方形（QGIS では常に長方形）です。典型的なラスタデータセットには、航空写真や衛星画像などのリモートセンシングデータ、標高や気温などのモデル化されたデータなどがあります。

ベクタデータとは異なり、ラスタデータは通常、各セルに関連するデータベースレコードを持ちません。これらのデータは、ピクセル解像度とラスタレイヤのコーナーピクセルの X/Y 座標によってジオコーディングされています。これにより、QGIS はマップキャンバス上にデータを正確に配置することができます。

GeoPackage 形式は、QGIS で作業する際にラスタデータを保持するのに便利な形式です。一般的で強力な機能を持つ GeoTiff 形式も良い選択肢になります。

QGIS は適切にデータを表示するために、ラスタレイヤ内部（例： GeoTiff ）や関連付けられた **ワールドファイル** のジオリファレンス情報を利用しています。

15.3.2 ベクタデータ

QGIS で利用できる機能やツールの多くは、ベクタデータのソースに関係なく同じように動作します。ただし、フォーマット（GeoPackage、ESRI シェープファイル、MapInfo や MicroStation ファイル形式、AutoCAD DXF、PostGIS、SpatiaLite、DB2、Oracle Spatial、MS SQL Server、SAP HANA 空間データベース、その他多数）の仕様が異なるため、QGIS では一部のプロパティの扱いが異なる場合があります。フォーマットのサポートは GDAL vector drivers によって提供されています。このセクションでは、この差異をどのように扱うかについて説明します。

注釈: QGIS ではポイント（マルチポイント）、ライン（マルチライン）、ポリゴン（マルチポリゴン）、円形ストリング、複合カーブ、カーブポリゴン、マルチカーブ、マルチサーフェスの地物タイプをサポートしており、全ての地物タイプはオプションで Z 値や M 値を持つことができます。

一部のドライバは円形ストリング、複合カーブ、カーブポリゴン、マルチカーブ、マルチサーフェスの地物タイプなどをサポートしていないことに注意してください。QGIS はこれらを変換します。

GeoPackage

GeoPackage （GPKG）形式はプラットフォームに依存しない形式で、SQLite データベースコンテナとして実装されており、ベクタデータとラスタデータの両方を格納するために使用することができます。この形式は Open Geospatial Consortium （OGC）によって定義され、2014 年に公開されました。

GeoPackage は SQLite データベース内に以下のものを格納するために使用できます：

- **ベクタ** 地物
- **画像のタイルマトリックスセット** や **ラスタ** マップ
- 属性データ（非空間データ）
- 拡張情報

QGIS バージョン 3.8 以降では、GeoPackage は QGIS のプロジェクトを保存できるようになりました。GeoPackage レイヤは JSON フィールドを持つことができます。

GeoPackage は QGIS においてベクタデータのデフォルトの形式です。

ESRI シェープファイル形式

例えば GeoPackage や SpatiaLite と比較するといくつかの制限はありますが、ESRI シェープファイル形式は未だに最も使われているベクタファイル形式の一つです。

ESRI シェープファイル形式のデータセットはいくつかのファイルからなります。下記の 3 つは必須です。

1. `.shp` ファイルは地物のジオメトリを持ちます。
2. `.dbf` ファイルは dBase 形式で属性を保持します。
3. `.shx` インデックスファイル

ESRI シェープファイル形式のデータセットは、投影法に関する情報を持った .prj 拡張子を持つファイルを含めることもできます。投影法ファイルがあると便利ですが、これは必須ではありません。シェープファイル形式のデータセットには追加のファイルを含めることもできます。詳細については、ESRI の 技術仕様 を参照してください。

GDAL では圧縮された ESRI シェープファイル形式（ shz や shp.zip ）の読み書きをサポートしています。

ESRI シェープファイル形式のデータセットのパフォーマンスを向上させる

ESRI シェープファイル形式のデータセットの描画パフォーマンスを向上するには、空間インデックスを作成します。空間インデックスによってズーム、パン両方の速度が向上します。QGIS で使用される空間インデックスは .qix 拡張子を持ちます。

空間インデックスを作成するには、以下の手順で操作します：

1. ESRI シェープファイル形式のデータセットを読み込む（ **ブラウザパネル** 参照）
2. 凡例内のレイヤ名をダブルクリックするか、右クリックしてコンテキストメニューから **プロパティ**... を選択し、 **レイヤプロパティ** ダイアログを開く
3. **ソース** タブで、 **空間インデックスの作成** ボタンを押す

.prj ファイルの読み込みに関する問題

.prj ファイルを持った ESRI シェープファイル形式のデータセットを読み込んでも QGIS がそのファイルから座標参照系を読み取れない場合は、レイヤの **レイヤプロパティ** **ソース** タブで CRS の選択 ボタンをクリックし、適切な投影法を手動で定義する必要があります。これは、 .prj ファイルが *CRS* ダイアログで表示されるような QGIS で使用される投影法パラメータを完全には指定していないことが多いためです。

同じ理由で、QGIS で新しい ESRI シェープファイル形式のデータセットを作成すると、2 つの異なる投影ファイルが作成されます：ESRI ソフトウェアと互換性のある、限定された投影パラメータの .prj ファイルと、CRS のすべてのパラメータを指定する .qpj ファイルです。QGIS が .qpj ファイルを見つけた時には、これが .prj ファイルの代わりに使用されます。

区切りテキストファイル

区切りテキストファイルは、そのシンプルさと読みやすさから非常に一般的であり広く使われています。データはプレーンテキストエディタで閲覧・編集が可能です。区切りテキストファイルは表形式のデータで、列は決められた文字で区切られ、行は改行で区切られます。通常、最初の行には列名が含まれています。区切りテキストファイルの一般的なタイプは、列がカンマで区切られた CSV（Comma Separated Values）です。区切りテキストファイルは位置情報を含むこともできます（ **区切りテキストファイルへのジオメトリ情報の保存** を参照してください）。

QGIS では、区切りテキストファイルをレイヤや通常のテーブルとして読み込むことができます（ **ブラウザパネル** や **区切りテキストファイルをインポートする** を参照）。まずは、ファイルが以下の条件を満たしていることを確認しましょう。

1. ファイルにはフィールド名を区切ったヘッダ行が必要です。これはデータの最初の行（理想的にはテキストファイルの最初の行）でなければなりません。
2. ジオメトリを有効にする場合には、ファイルにはジオメトリを定義するフィールドが含まれている必要があります。これらのフィールドには任意の名前を付けることができます。

3. （ジオメトリが座標で定義されている場合は）X と Y の座標フィールドは数値で指定する必要があります。座標系は重要ではありません。

4. 文字列以外の列を持つ CSV ファイルがある場合、付随する CSVT ファイルを持つことができます（*CSVT* **ファイルを使用したフィールドの書式制御** のセクションを参照してください）。

QGIS サンプルデータセット（**サンプルデータのダウンロード** のセクション参照）内の標高点データファイル `elevp.csv` は、有効なテキストファイルの例です。

```
X;Y;ELEV
-300120;7689960;13
-654360;7562040;52
1640;7512840;3
[...]
```

このテキストファイルの注意点をいくつか挙げます：

1. このテキストファイル例では、 `;`（セミコロン）を区切り文字として使用しています（フィールドの区切りには任意の文字を使用可能です）。

2. 最初の行は見出しです。ここには `X` 、 `Y` および `ELEV` というフィールドが含まれています。

3. テキストフィールドを区切るための引用符（ `"` ）は使われていません。

4. X 座標は `X` フィールドに含まれています

5. Y 座標は `Y` フィールドに含まれています

区切りテキストファイルへのジオメトリ情報の保存

区切りテキストファイルには、主に 2 つの形式でジオメトリ情報を含めることができます：

- ポイントジオメトリデータの場合は、別々の列に座標（例えば `Xcol` 、 `Ycol...` ）として含める
- 任意のジオメトリタイプの場合は、単一の列にジオメトリの well-known text (WKT) 表現として含める

曲線ジオメトリ (CircularString, CurvePolygon, CompoundCurve) を持つ地物がサポートされています。以下は、WKT として符号化されたジオメトリを持った区切りテキストファイルに含まれるジオメトリタイプの例です:

```
Label;WKT_geom
LineString;LINESTRING(10.0 20.0, 11.0 21.0, 13.0 25.5)
CircularString;CIRCULARSTRING(268 415,227 505,227 406)
CurvePolygon;CURVEPOLYGON(CIRCULARSTRING(1 3, 3 5, 4 7, 7 3, 1 3))
CompoundCurve;COMPOUNDCURVE((5 3, 5 13), CIRCULARSTRING(5 13, 7 15,
  9 13), (9 13, 9 3), CIRCULARSTRING(9 3, 7 1, 5 3))
```

区切りテキストファイルはジオメトリの Z 座標や M 座標もサポートしています:

```
LINESTRINGZ(10.0 20.0 30.0, 11.0 21.0 31.0, 11.0 22.0 30.0)
```

CSVT ファイルを使用したフィールドの書式制御

CSV ファイルを読み込む際 GDAL ドライバは、別の指定がない限り、全てのフィールドを文字列（テキスト）と仮定します。CSVT ファイルを作成することで、さまざまな列のデータ型を GDAL（そして QGIS）に伝えることができます。

タイプ	名前	例
整数値	Integer	4
ブール値	Integer(Boolean)	true
小数点付き数値	Real	3.456
日付	Date (YYYY-MM-DD)	2016-07-28
時刻	Time (HH:MM:SS+nn)	18:33:12+00
日付と時刻	DateTime (YYYY-MM-DD HH:MM:SS+nn)	2016-07-28 18:33:12+00
CoordX	CoordX	8.8249
CoordY	CoordY	47.2274
Point(X)	Point(X)	8.8249
Point(Y)	Point(Y)	47.2274
WKT	WKT	POINT(15 20)

CSVT ファイルは、データ型が引用符で囲まれ、コンマで区切られた **一行** のプレーンテキストファイルです。例えば

```
"Integer","Real","String"
```

各列の幅や精度を指定することもできます。例

```
"Integer(6)","Real(5.5)","String(22)"
```

このファイルは `.csv` ファイルと同じフォルダに同じ名前で保存されますが、拡張子は `.csvt` です。

更なる情報については GDAL CSV Driver を参照してください。

Tip: フィールド型を検知する

CSVT ファイルを使用してデータ型を伝える代わりに、QGIS はフィールド型を自動的に検出し、想定されるフィールド型を変更する可能性を提供します。

PostGIS レイヤ

PostGIS レイヤは PostgreSQL データベースに保存されます。PostGIS の利点は、空間インデックス、フィルタリング、クエリする能力です。PostGIS を使用すると、select や identify などのベクタ関数は、QGIS の GDAL レイヤを使用した場合よりも正確に動作します。

Tip: PostGIS レイヤ

通常、PostGIS レイヤは geometry_columns テーブルのエントリがあることによって識別されます。QGIS では、geometry_columns テーブルのエントリを持たないレイヤもロードすることができます。これには、テーブルとビューの両方が含まれます。ビューの作成については、PostgreSQL のマニュアルを参照してください。

このセクションでは、QGIS が PostgreSQL レイヤにアクセスする方法について、いくつかの詳細を説明します。ほとんどの場合、QGIS は単純にロード可能なデータベーステーブルのリストをユーザーに提供し、要求に応じてテーブルを読み込むだけです。しかし、PostgreSQL のテーブルを QGIS に読み込むことができない場合には、以下の情報が QGIS のメッセージを理解するための助けとなり、PostgreSQL のテーブルやビュー定義を修正して QGIS で読み込めるようになるための方向性が得られるかもしれません。

注釈: PostgreSQL データベースにも QGIS のプロジェクトを保存できます。

主キー

QGIS は、PostgreSQL レイヤがユニークなキーとして使用できる列を含んでいることを要求します。テーブルの場合、これは通常、テーブルに主キーまたはユニーク制約を持つ列を必要とすることを意味します。QGIS では、この列は int4 型（4 バイトサイズの整数）である必要があります。または、ctid 列を主キーとして使用することもできます。テーブルにこれらの項目がない場合、oid 列が代わりに使用されます。列にインデックスが作成されると、パフォーマンスが向上します（PostgreSQL では主キーが自動的にインデックス化されることに注意してください）。

QGIS には、デフォルトで有効になっている **id で選択** チェックボックスがあります。このオプションは属性なしの id を取得しますが、多くの場合、これは主キーによる検索よりも高速です。

ビュー

PostgreSQL のレイヤがビューである場合でも同様の要件がありますが、ビューは必ずしも主キーやユニーク制約のある列を持っているとは限りません。ビューをロードする前に、QGIS ダイアログで主キーフィールド（整数である必要があります）を定義する必要があります。適切な列がビューに存在しない場合、QGIS はレイヤをロードしません。このような場合には、適切な列（整数型で、主キーであるかまたはユニーク制約があり、できればインデックス付き）を確実に含むようにビューを変更することが解決策となります。

テーブルに関しては、**id で選択** のチェックボックスがデフォルトで有効になっています（チェックボックスの意味については上記参照）。複雑なビューを使用する場合には、このオプションを無効にした方がよいかもしれません。

注釈: PostgreSQL 外部テーブル

PostgreSQL 外部テーブルは PostgreSQL プロバイダによって明示的にサポートされてはおらず、ビューのように扱われます。

QGIS layer_style テーブルとデータベースのバックアップ

`pg_dump` コマンドと `pg_restore` コマンドを使用して PostGIS データベースのバックアップを作成したい場合、QGIS で保存されたデフォルトのレイヤスタイルがその後のリストアに失敗する場合には、リストアコマンドの前に XML オプションを `DOCUMENT` に設定する必要があります。

1. `layer_style` テーブルの PLAIN バックアップを作ります
2. そのファイルをテキストエディタで開きます
3. `SET xmloption = context;` の行を `SET XML OPTION DOCUMENT;` に変えます
4. ファイルを保存します
5. psql を使ってテーブルを新しいデータベースにリストアします

データベース側でのフィルタ

QGIS では、サーバサイドであらかじめ地物をフィルタリングすることができます。 **設定　オプション　データソース**　の **可能であればサーバ側で式を実行する** にチェックを入れると、サーバ側でフィルタリングされます。データベースに送信されるのは、サポートされている式のみです。サポートされていない演算子や関数を使用した式は、ローカル側での評価に滑らかにフォールバックします。

PostgreSQL のデータ型のサポート

PostgreSQL プロバイダでサポートされているデータ型には、以下のものが含まれます：整数、浮動小数点、論理値、バイナリオブジェクト、可変長文字、幾何データ、 日付/時刻、配列、 hstore（キー/バリュー型）、JSON

PostgreSQL へデータをインポートする

データは、DB マネージャプラグインやコマンドラインツールの shp2pgsql、ogr2ogr など、いくつかのツールを使用して PostgreSQL/ PostGIS にインポートできます。

DB マネージャ

QGIS には、  DB マネージャ という名前のコアプラグインが付属しています。これはデータをロードするために使用することができ、スキーマもサポートしています。詳細は *DB* **マネージャプラグイン** を参照してください。

shp2pgsql

PostGIS には **shp2pgsql** というユーティリティコマンドが含まれており、これを使用してシェープファイル形式のデータセットを PostGIS 対応のデータベースにインポートすることができます。例えば、`lakes.shp` という名前のシェープファイル形式のデータセットを `gis_data` という名前の PostgreSQL データベースにインポートするには、次のコマンドを使用します:

```
shp2pgsql -s 2964 lakes.shp lakes_new | psql gis_data
```

これにより、 `gis_data` データベース内に `lakes_new` という名前の新規レイヤが作成されます。この新しいレイヤは空間参照識別子（SRID）が 2964 になります。空間参照系と投影法についての詳細は、 **投影法の利用方法** を参照してください。

Tip: PostGIS からデータセットをエクスポートする

PostGIS のデータセットをシェープファイル形式へとエクスポートするツール **pgsql2shp** もあります。このツールは PostGIS ディストリビューションに付属しています。

---

ogr2ogr

shp2pgsql や **DB マネージャ** の他にも、PostGIS にデータを読み込ませるツールがあります。 **ogr2ogr** です。これは GDAL のインストールに含まれています。

シェープファイル形式のデータセットを PostGIS にインポートするには、次のコマンドを使用します:

```
ogr2ogr -f "PostgreSQL" PG:"dbname=postgis host=myhost.de user=postgres
password=topsecret" alaska.shp
```

このコマンドは、シェープファイル形式のデータセット `alaska.shp` をホストサーバー *myhost.de* 上の PostGIS データベース *postgis* にユーザー *postgres* 、パスワード *topsecret* を使用してインポートします。

GDAL は PostGIS をサポートするために PostgreSQL と一緒にビルドされなければならないことに注意してください。(|nix|で) 入力することでこれを確認できます:

```
ogrinfo --formats | grep -i post
```

デフォルトの **INSERT INTO** メソッドの代わりに PostgreSQL の **COPY** コマンドを使用したい場合には、以下の環境変数を設定します（ と では利用可能）:

```
export PG_USE_COPY=YES
```

shp2pgsl は空間インデックスを作成しますが、 **ogr2ogr** では空間インデックスを作成しません。インポート後に追加のステップとして、通常の SQL コマンド **CREATE INDEX** を使用して手動でインデックスを作成する必要があります（次のセクション **パフォーマンスの改善** で説明します）。

パフォーマンスの改善

PostgreSQL データベースからの地物の取得は、特にネットワーク経由では時間がかかります。データベース内の各レイヤに PostGIS 空間インデックスがあることで、PostgreSQL レイヤの描画パフォーマンスを向上させることができます。PostGIS は空間検索を高速化するために GiST (Generalized Search Tree) インデックスの作成をサポートしています（GiST インデックスの情報は https://postgis.net にある PostGIS ドキュメントから引用しています）。

Tip: レイヤのインデックス作成には DB マネージャを使用することができます。最初にレイヤを選択し、**テーブル　テーブルの編集** をクリックして、 **インデックス** タブを開き、 **空間インデックスの追加** ボタンを押します。

GiST インデックスを作成するための構文は以下のとおりです:

```
CREATE INDEX [indexname] ON [tablename]
  USING GIST ( [geometryfield] GIST_GEOMETRY_OPS );
```

巨大なテーブルでは、インデックスの作成には長時間かかることに注意してください。インデックスが作成されたら、 VACUUM ANALYZE を実行してください。詳細な情報については、PostGIS ドキュメント（ **文献と** *Web* **参照** の POSTGIS-PROJECT ）を参照してください。

下記の例では、GiST インデックスを作成しています:

```
gsherman@madison:~/current$ psql gis_data
Welcome to psql 8.3.0, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

gis_data=# CREATE INDEX sidx_alaska_lakes ON alaska_lakes
gis_data-# USING GIST (the_geom GIST_GEOMETRY_OPS);
CREATE INDEX
gis_data=# VACUUM ANALYZE alaska_lakes;
VACUUM
gis_data=# \q
gsherman@madison:~/current$
```

SpatiaLite レイヤ

ベクタレイヤを SpatiaLite 形式で保存したい場合には、**既存のレイヤから新しいレイヤを作成する** の説明に従ってください。**形式** として `SpatiaLite` を選択し、**ファイル名** と **レイヤ名** を入力してください。

また、形式として `SQLite` を選択し、**カスタムオプション** **データソース** フィールドに ``SPATIALITE=YES``を追加することでも SpatiaLite 形式で保存できます。このカスタムオプションは、GDAL に SpatiaLite データベースを作成するように指示します。 https://gdal.org/drivers/vector/sqlite.html も参照してください。

QGIS は SpatiaLite で編集可能なビューもサポートしています。SpatiaLite データの管理には、*DB* **マネージャ** コアプラグインを使用することもできます。

新しい SpatiaLite レイヤを作成したい場合は **新しい** *SpatiaLite* **レイヤを作成する** を参照して下さい。

GeoJSON に特有のパラメータ

GeoJSON 形式で **レイヤをエクスポートする** 場合、いくつか特有の **レイヤオプション** があります。これらのオプションは、ファイルに書き出す GDAL のオプションに由来するものです：

- *COORDINATE_PRECISION* 座標を書き込む際の小数点区切り文字以下の最大桁数を指定します。デフォルトは 15 です（注：緯度経度座標の場合には 6 で十分です）。末尾に続くゼロは切り捨てられます。
- *RFC7946* デフォルトでは GeoJSON 2008 が使用されます。この値が YES に設定されると、更新された RFC 7946 標準が使用されます。デフォルトは NO です（従って GeoJSON 2008 が使用されます）。主な違いについては https://gdal.org/drivers/vector/geojson.html#rfc-7946-write-support を参照してください。簡単に言えば、RFC 7946 標準では EPSG:4326 のみが許可され、他の CRS は変換されます。また、ポリゴンは向きの右手ルールに従うように書かれ、"bbox" 配列の値は [west, south, east, north] であり、[minx, miny, maxx, maxy] ではありません。そして、FeatureCollection、Feature、Geometry オブジェクトではいくつかの拡張メンバ名が禁止されており、デフォルトの座標精度は 10 進数で 7 桁です。
- *WRITE_BBOX* を YES に設定すると、地物や地物コレクションのレベルでジオメトリのバウンディングボックスを含めます。

GeoJSON の他に、"GeoJSON - Newline Delimited" （ https://gdal.org/drivers/vector/geojsonseq.html 参照）へと出力するオプションもあります。フィーチャーコレクションのフィーチャーの代わりに、1 種類のもの（おそらくは単一のフィーチャー）を改行区切りで連続的に並べます。

GeoJSON - Newline Delimited にもいくつかの特有のレイヤオプションがあります。

- *COORDINATE_PRECISION* 上記を参照してください（GeoJSON のものと同様です）。
- *RS* レコードの開始を RS=0x1E の文字とするかどうかを決めます。違いは、フィーチャーがどのように区切られるかにあります。 改行（LF）文字のみ（Newline Delimited JSON, geojsonl）か、レコード区切り（RS）文字がレコードの前に置かれて改行文字で区切られるか（GeoJSON Text Sequences, geojsons）です。 デフォルトは NO です。拡張子が指定されない場合は、ファイルに `.json` 拡張子が付きます。

SAP HANA 空間レイヤ

このセクションでは、QGIS が SAP HANA のレイヤにアクセスする方法について、いくつかの詳細を説明します。ほとんどの場合、QGIS は単純にロード可能なデータベーステーブルのリストをユーザーに提供し、要求に応じてテーブルを読み込むだけです。しかし、SAP HANA のテーブルまたはビューを QGIS に読み込むことができない場合には、以下の情報が参考となり、根本的な原因を理解し問題解決の助けとなるでしょう。

地物の識別

QGIS の地物編集機能の全てを利用したい場合には、QGIS がレイヤ内の各地物を明確に識別できる必要があります。内部的には、QGIS は地物の識別に 64 ビットの符号付き整数を使用します。負の値の範囲は特別な目的のために予約されています。

従って、QGIS の地物編集機能を完全にサポートするためには、SAP HANA プロバイダには 64 ビットの正の整数にマッピングできるユニークキーが必要です。もしもこのようなマッピングを作成できない場合には、地物を表示することはできますが、編集できない可能性があります。

テーブルの追加

レイヤとしてテーブルを追加する際に、SAP HANA プロバイダはテーブルの主キーを使用して、主キーをユニークな地物 ID にマッピングします。 従って、地物編集機能を完全にサポートするためには、テーブルの定義に主キーが必要です。

SAP HANA プロバイダはマルチカラムの主キーをサポートしていますが、ベストな性能を得たい場合には、主キーは `INTEGER` 型の単一カラムとしてください。

ビューの追加

レイヤとしてビューを追加する際に、SAP HANA プロバイダは地物を一義的に特定するカラムを自動的に特定することはありません。さらに、一部のビューは読み取り専用で、編集できません。

地物編集機能を完全にサポートするためには、ビューは更新可能である必要があり（システムビュー `SYS.VIEWS` において、問題となっているビューのカラムの `IS_READ_ONLY` を確認してください）、地物を識別するための一つ以上のカラムを QGIS で手動で指定する必要があります。このカラムは、 **レイヤ　レイヤを追加　*SAP HANA* 空間レイヤを追加** を使用して、 **地物** *ID* カラムで選択することで指定できます。ベストな性能を得たい場合には、 **地物** *ID* の値は単一の `INTEGER` 型のカラムとしてください。

15.3.3 経度 180 ° をまたぐレイヤ

多くの GIS パッケージは、地理参照系（緯度経度）が 180 度の経線を越えるレイヤをラップしません。その結果、このようなレイヤを QGIS で開くと、近くに表示されるはずの 2 つの場所が大きく離れて表示されることがあります。図 15.27 では、マップキャンバスの左端にある小さな点（チャタム諸島）は、ニュージーランド本島の右側のグリッド内にあるはずです。

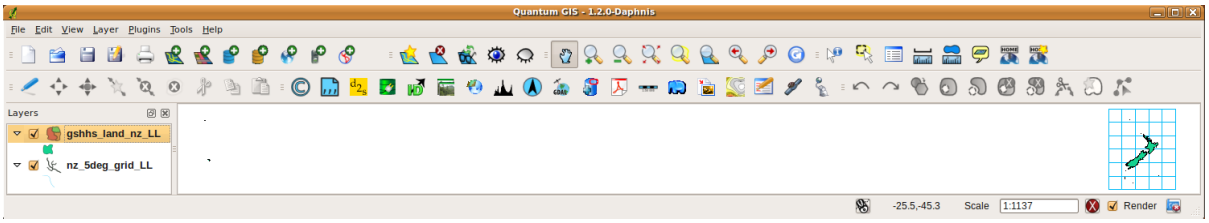


図 15.27: 180 ° 経線をまたぐ緯度経度による地図

PostGIS で解決する

回避策として、PostGIS と ST_ShiftLongitude 関数を使用して経度値を変換する方法があります。この関数は、ジオメトリ内のすべての地物のすべてのコンポーネントのすべての点/頂点を読み取り、その経度座標を -180...0 ° から 180...360 ° に、そしてその逆にシフトします。この関数は対称であるため、結果は-180...180 ° のデータを 0...360 ° で表現し、0...360 ° のデータを-180...180 ° で表現します。

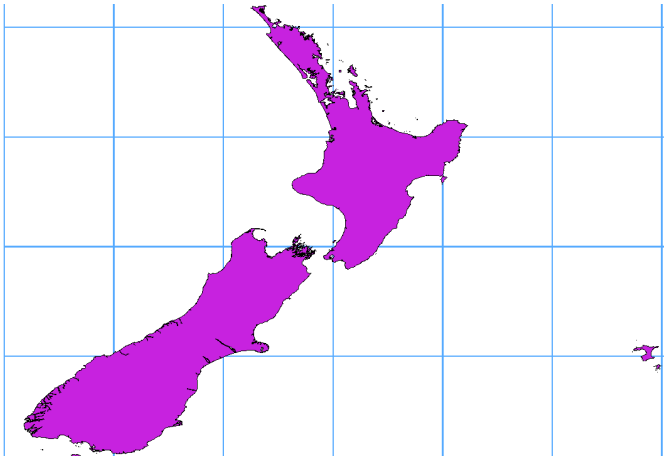


図 15.28: **ST_ShiftLongitude** 関数を適用した経度 180 ° 越え

1. 例えば DB マネージャプラグインを使用して、データを PostGIS にインポートします（ *PostgreSQL へデータをインポートする* 参照）。
2. PostGIS のコマンドラインインターフェイスを使って、以下のコマンドを実行します：

```
-- In this example, "TABLE" is the actual name of your PostGIS table
update TABLE set geom=ST_ShiftLongitude(geom);
```

3. すべてうまくいくと、更新された地物の数についての確認が表示されます。その後、地図を読み込むと違いを確認できます（ *Figure_vector_crossing_map* ）。

第16章　ベクタデータの操作

16.1 ベクタプロパティダイアログ

ベクタレイヤの **レイヤプロパティ** ダイアログには、地図内のレイヤの地物の外観（シンボロジ、ラベル、ダイアグラム）やマウスとのインタラクション（アクション、地図の Tips、フォームデザイン）を管理するための一般的な設定があります。また、レイヤについての情報を提供します。

レイヤプロパティ ダイアログにアクセスするには：

- **レイヤ** パネル内において、レイヤをダブルクリックするか、レイヤを右クリックしてポップアップメニューから **プロパティ**... を選択する
- レイヤを選択した状態で、 **レイヤ**　**レイヤのプロパティ**... メニューを選ぶ

ベクタ **レイヤプロパティ** ダイアログには以下のセクションがあります：

情報	ソース	シンボロジ [1]
ラベル [1]	マスク [1]	*3D* ビュー [1]
ダイアグラム	属性	属性フォーム
テーブル結合	補助テーブル	アクション
表示名	レンダリング	時系列
変数	標高	メタデータ
依存関係	凡例	*QGIS* サーバー
デジタイズ	外部プラグイン [2] タブ	

[1] **レイヤスタイルパネル** からも利用可能です

[2] インストールした **外部プラグイン** は、任意でこのダイアログにタブを追加することがあります。これらについては、このドキュメントでは説明しません。外部プラグインのドキュメントを参照してください。

Tip: レイヤスタイルの完全なまたは部分的なプロパティを共有する

ダイアログの下部にある **スタイル** メニューを使用すると、レイヤスタイルのプロパティまたはその一部をさまざまな対象（ファイル、クリップボード、データベース）からインポートしたり、対象へエクスポートしたりすることができます。 **カスタムスタイルを管理する** を参照してください。

注釈: 埋め込まれたレイヤ（ **外部プロジェクトからのレイヤの埋め込み** を参照）のプロパティ（シンボロジ、ラベル、アクション、デフォルト値、フォーム...）は、元のプロジェクトファイルから引用されているため、この動作を壊す可能性のある変更を避けるために、埋め込まれたレイヤに対してはレイヤプロパティダイアログは利用できなくなっています。

16.1.1 情報プロパティ

情報 タブは読み取り専用で、現在のレイヤの要約された情報やメタデータをさっと掴むことができる興味深い場所です。提供される情報には、以下のものがあります：

- 一般情報：プロジェクト内での名前、ソースへのパス、付随的なファイルのリスト、最終更新時刻、ファイルの大きさ、使用しているプロバイダ
- レイヤのプロバイダからの情報：ストレージ形式、ジオメトリタイプ、データソースの文字コード、領域、地物数など
- 空間参照システム（CRS）：CRS の名前、単位、投影法、精度、参照（静的か動的か）
- **入力されたメタデータ** からの情報：アクセス、領域、リンク、連絡先、履歴など
- ジオメトリ（空間的な範囲や CRS など）や、属性（フィールド数や各フィールドの特性など）に関連した情報

16.1.2 ソースプロパティ

このタブでは、ベクタレイヤの一般的な設定の定義ができます。

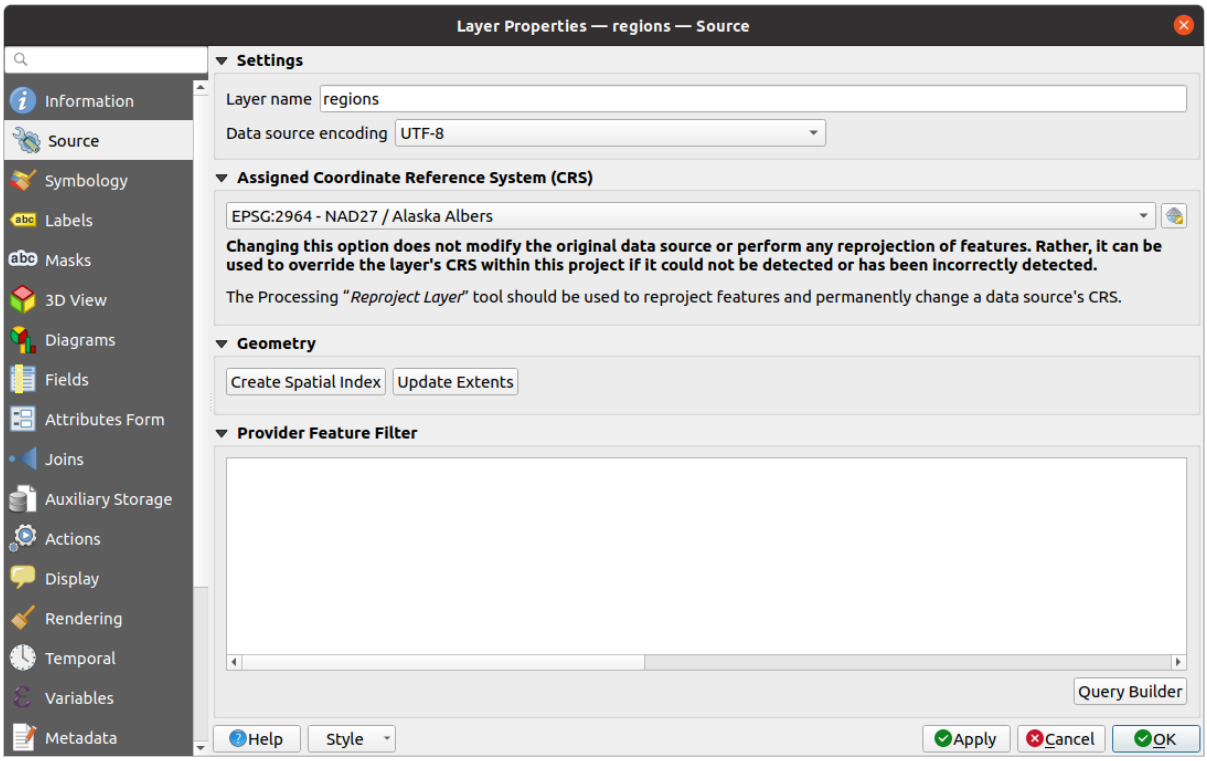


図 16.1: ベクタレイヤプロパティダイアログのソースタブ

設定

- レイヤのファイル名とは異なる **レイヤ名** を設定すると、プロジェクト内（ **レイヤパネル** や式、印刷レイアウトの凡例など）でレイヤを識別するために使用されます。
- データフォーマットにもよりますが、QGIS が検出した文字コードが正しくない場合に **データソースの文字コード** を選択します。

CRS とジオメトリ

- レイヤに **設定された** *CRS* が表示されます。最近使用した CRS をドロップダウンリストから選ぶか、CRS の選択 ボタン（ **座標参照系セレクタ** 参照）をクリックすることで、レイヤの CRS を変更できます。レイヤに適用された CRS が間違っていた場合か、CRS が何も適用されていない場合にのみ、この操作を行ってください。データを別の CRS に再投影したい場合には、プロセシングアルゴリズムのレイヤの再投影を使用するか、 **別のレイヤに保存** してください。
- **空間インデックスの作成** （OGR がサポートする形式のみです）
- **領域の更新** レイヤの領域情報を更新します。

クエリビルダ

クエリビルダ ダイアログは、レイヤプロパティダイアログの **ソース** タブの下部、 **プロバイダ地物フィルタ** グループの下にある名前どおりのボタンからアクセスできます。

クエリビルダは、SQL ライクな WHERE 句を使用してレイヤ内の地物のサブセットを定義し、メインウィンドウにその結果を表示するためのインターフェースを提供します。クエリが有効である限り、プロジェクトで利用可能となる地物はクエリの結果に対応するもののみです。

クエリビルダ でフィルタを定義するために、複数のレイヤ属性を使用することができます。複数の属性の使用例を 図 16.2 に示します。この例では、フィルタは属性の結合を行っています。

- `toa` (DateTime フィールド: `cast("toa" as character) > '2017-05-17'` 、 `cast("toa" as character) < '2019-12-24T18:00:00'`)、
- `name` (String フィールド: `"name" > 'S'`)、
- `FID` (Integer フィールド: `FID > 10`)

これらを AND、OR そして NOT 演算子と括弧で結合しています。この文法（ `toa` フィールドの DateTime 形式も含め）は、GeoPackage データセットで動作します。

フィルタはデータプロバイダ（OGR、PostgreSQL、MS SQL Server...）レベルで行われます。そのため、構文はデータプロバイダに依存します（例えば、DateTime は ESRI Shapefile フォーマットではサポートされていません）。完全な式:

```
cast("toa" as character) > '2017-05-17' AND
cast("toa" as character) < '2019-12-24T18:00:00' AND
NOT ("name" > 'S' OR FID > 10)
```

また、 **クエリビルダ** ダイアログは **レイヤ** メニューまたはレイヤのコンテキストメニューの **フィルタ...** オプションを使用して開くこともできます。ダイアログの **属性** 、 **値** 、 **演算子** セクションは、 **プロバイダ特有のフィルタ式** ボックスに表示される SQL ライクなクエリを構築するのに役立ちます。

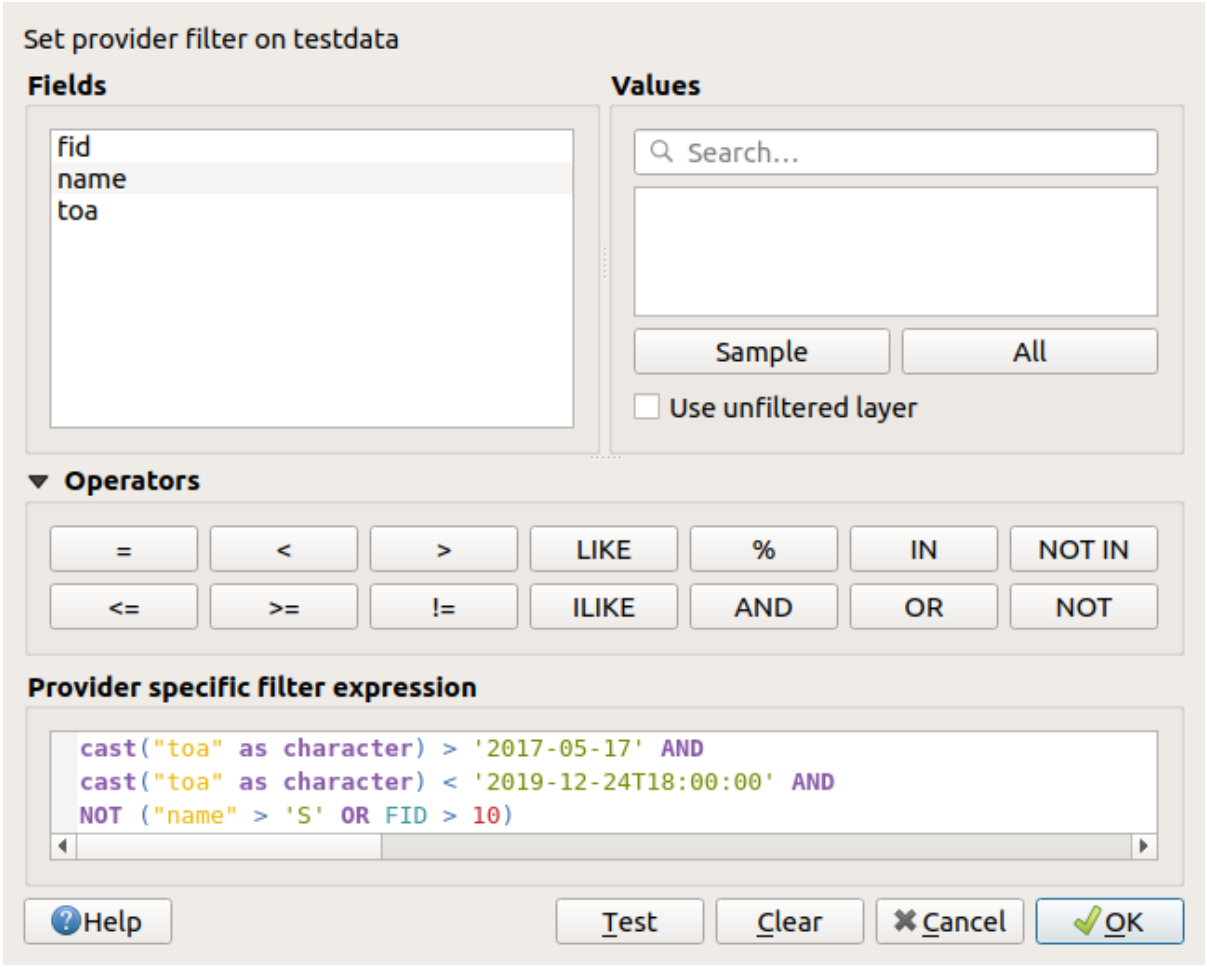


図 16.2: クエリビルダ

属性 のリストには、レイヤの全ての属性が含まれます。SQL WHERE 句フィールドに属性カラムを追加するには、属性名をダブルクリックするか、SQL ボックスに属性名を入力します。

値 フレームは、現在選択されている属性の値をリストします。属性のユニークな値を全てリストするには、**全て** ボタンをクリックします。代わりに、カラムのユニークな値を最初の 25 個リストするには、**サンプル** ボタンをクリックします。SQL WHERE 句フィールドに値を追加するには、値リストの値をダブルクリックします。値フレームの上部にある検索ボックスを使用して、リスト内から属性値を簡単にブラウズして見つけることができます。

演算子 セクションには、使用可能な演算子全てが含まれています。SQL WHERE 句フィールドに演算子を追加するには、適切なボタンをクリックします。関係演算子（ = 、 > など...）、文字列比較演算子（ LIKE ）、論理演算子（ AND 、 OR など...）が利用可能です。

テスト ボタンはクエリをチェックするのに役立ち、現在のクエリを満足する地物の数がメッセージボックスに表示されます。 **クリア** ボタンを使うと、SQL クエリを消去し、レイヤを元の状態（すなわち全ての地物を完全に読み込む状態）に戻します。

フィルタが適用されている場合、QGIS は結果として得られるサブセットがレイヤの全体であるかのように扱います。例えば、 'Borough' でフィルタ（ "TYPE_2" = 'Borough' ）を適用すると、 Anchorage を表示することも、クエリすることも、保存も編集もできません。なぜならば、これは 'Municipality' であり、従ってサブセットの部分には含まれないからです。

Tip: レイヤパネル内におけるフィルタされたレイヤの表示

レイヤ パネルにおいて、フィルタされたレイヤはレイヤの横に  フィルター アイコンが表示され、ボタンの上にマウスを置くと、使用されているクエリが表示されます。アイコンをダブルクリックすると **クエリビルダ** ダイアログが開き、フィルタを編集できます。

16.1.3 シンボロジプロパティ

 シンボロジタブには、ベクタデータのレンダリングとシンボライズのための包括的なツールが用意されています。すべてのベクタデータに共通のツールや、さまざまな種類のベクタデータ用に作られた特別なシンボライズツールを使用することができます。ただし、すべてのタイプで次のようなダイアログ構造は共通です：上部には分類と地物のシンボルを準備するのに役立つウィジェットがあり、下部には **レイヤレンダリング** ウィジェットがあります。

Tip: さまざまなレイヤ表現を素早く切り替える

レイヤプロパティ ダイアログの下部にある **スタイル** **追加** メニューを使用すると、必要なだけスタイルを保存できます。スタイルとは、レイヤの全てのプロパティ（例えばシンボロジ、ラベル、ダイアグラム、属性フォーム、アクションなど）を好きなように組み合わせたものです。そして、 **レイヤパネル** 内のレイヤのコンテキストメニューからスタイルを切り替えるだけで、異なるデータ表現が自動的に得られます。

Tip: ベクタシンボロジをエクスポートする

QGIS から Google *.kml や *.dxf、MapInfo *.tab ファイルにベクタシンボロジをエクスポートするオプションがあります。まずはレイヤの右マウスメニューを開き、 **名前を付けて保存...** をクリックして、出力ファイルの名前とその形式を指定します。ここでダイアログで **シンボロジのエクスポート** メニューを使用して、シンボルを **地物シンボロジ** または **シンボルレイヤシンボロジ** のいずれかとして保存します。シンボルレイヤを使用している場合には、2 番目の設定を使用することをお勧めします。

地物のレンダリング

レンダラーは、地物を正しいシンボルと共に描画する役割を持ちます。レイヤのジオメトリタイプに関係なく、一般的なレンダラーには、単一シンボル、カテゴリ値、連続値、ルールベースの 4 つのタイプがあります。ポイントレイヤーには、点の競合回避、点のクラスタ、ヒートマップレンダラーがあり、ポリゴンレイヤーには、結合済み地物、反転ポリゴン、2.5D レンダラーがあります。

連続色のレンダラーはありません。これは、実際には連続値レンダラーの特殊な場合でしかないからです。カテゴリ値レンダラーおよび連続値レンダラーは、シンボルとカラーランプを指定して作成することができ、これによってシンボルの色を適切に設定できます。各データタイプ（ポイント、ライン、およびポリゴン）には、ベクタシンボルレイヤタイプが利用可能です。選択したレンダラーによっては、ダイアログにさまざまな追加のセクションがあります。

注釈: ベクタレイヤのスタイルを設定する際にレンダラーの種類を変更しても、シンボルに設定した内容は維持されます。この機能は変更 1 回だけにしか働かないことに注意してください。レンダラーの種類の変更を繰り返すと、シンボルの設定は失われます。

単一シンボルレンダラー

単一定義（*single*）レンダラーは、単一のユーザー定義のシンボルを使用してレイヤの全ての地物をレンダリングするために使用します。シンボル表現についての情報は、 **シンボルセレクタ** を参照してください。

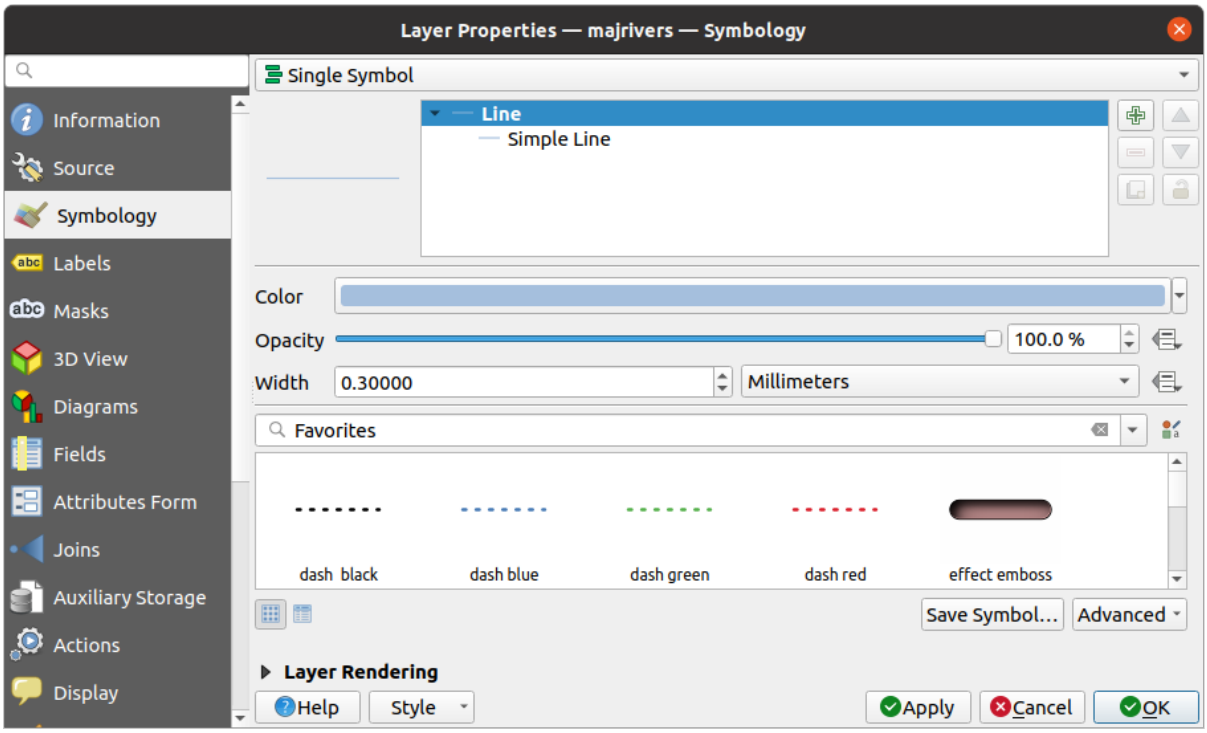


図 16.3: 単一シンボルラインプロパティ

シンボルなしレンダラー

シンボルなし レンダラーは、全ての地物に同一のレンダリングを適用することから、「単一定義シンボル」の特別なユースケースです。このレンダラーを使用すると、地物には何のシンボルも描画されませんが、ラベルやダイアグラム、その他の非シンボル部分は表示されます。

地物選択もキャンバスのレイヤで行うことができ、選択された地物はデフォルトのシンボルでレンダリングされます。編集中の地物も表示されます。

このレンダラーは、ラベルやダイアグラムだけを表示したいレイヤのための便利なショートカットです。この目的のために完全に透明な塗りつぶしやストロークでシンボルをレンダリングする必要はありません。

カテゴリ値レンダラー

カテゴリ値による定義（*categorized*） レンダラーは、属性値または式の離散的な値を反映する特性を持ったユーザー定義のシンボルを使用してレイヤの地物をレンダリングするために使用します。

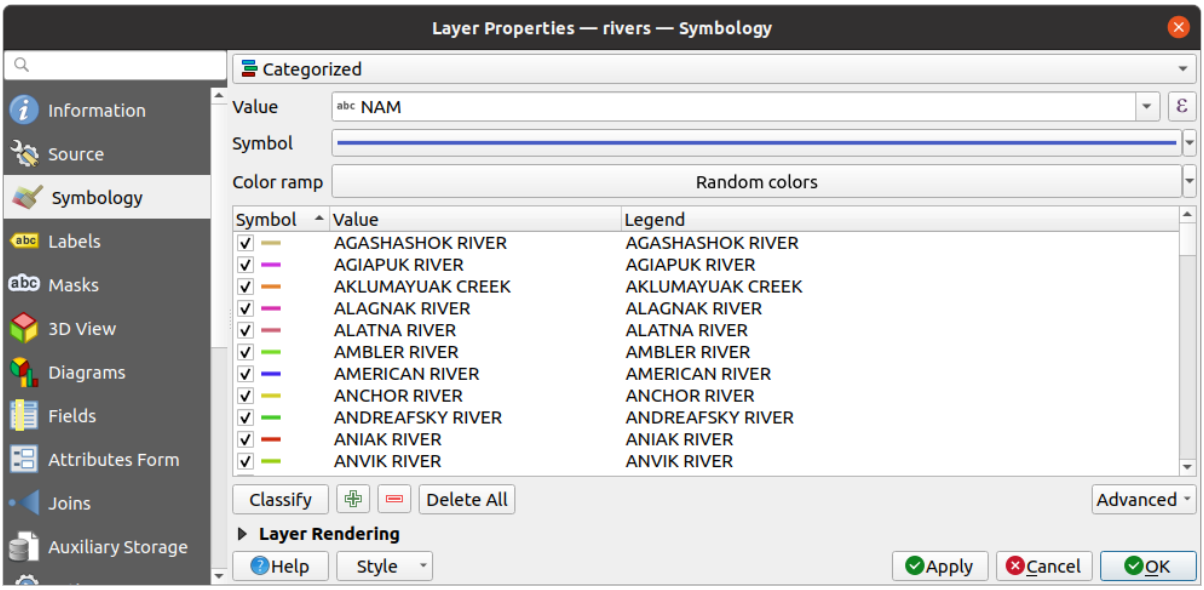


図 16.4: カテゴリ値シンボルのオプション

カテゴリ値によるシンボロジをレイヤに使用するには、以下のように操作します。

1. 分類の **値** の選択： これには既存のフィールドまたは **式** が使用できます。式はボックスに入力するか、関連する ε ボタンを使用して構築することができます。 カテゴリ値に式を使用することで、シンボロジのためにフィールドを作成する必要がなくなります（例えば、分類基準が複数の属性値に由来する場合など）。

   地物の分類のために使用する式は、どんな種類のものでも可能です。例えば、

   - 比較式：この場合、QGIS は `1` （ **True** ） か `0` （ **False** ）を返します。 以下は一例です。

     ```
     myfield >= 100
     $id = @atlas_featureid
     myfield % 2 = 0
     within( $geometry, @atlas_geometry )
     ```

   - 異なるフィールドの結合：

     ```
     concat( field_1, ' ', field_2 )
     ```

   - フィールド値を使った計算結果：

     ```
     myfield % 2
     year( myfield )
     ```

(次のページに続く)

(前のページからの続き)

```
field_1 + field_2
substr( field_1, -3 )
```

- 連続値から離散的なクラスへの変換：例えば、

```
CASE WHEN x > 1000 THEN 'Big' ELSE 'Small' END
```

- 複数の離散値を結合して単一カテゴリ化：例えば、

```
CASE
WHEN building IN ('residence', 'mobile home') THEN 'residential'
WHEN building IN ('commercial', 'industrial') THEN 'Commercial and Industrial
↪'
END
```

Tip: 地物を分類するためにはどんな種類の式も使用できますが、いくつかの複雑な式については **ルールによる定義に基づくレンダリング** を使用する方が簡単かもしれません。

---

2. 全てのクラスのベースシンボルとなる **シンボル** を設定します。
3. **カラーランプ** を指定します。すなわち、各シンボルに適用する色によって、色の範囲を選択します。

   カラーランプウィジェット の一般的なオプションに加えて、 **ランダムカラーランプ** をカテゴリに適用することもできます。色が気に入らない場合には、 **色をシャッフル** エントリをクリックすると新しいランダムカラーの組み合わせを再生成することができます。
4. 次に、 **分類** ボタンをクリックすると、与えられたフィールドもしくは式の離散値からクラスを生成します。
5. **ライブ更新** が利用できない場合には、 **適用** ボタンを押して変更を適用すると、マップキャンバス上の各地物がその分類のシンボルでレンダリングされます。

   デフォルトでは、QGIS は **その他の値** クラスをリストに追加します。初期状態では値が空ですが、このクラスは他のクラスに該当しない地物のためのデフォルトのクラスとして使用されます（例えば、分類フィールド/式にない値を持った新しい地物を作成した場合）。

デフォルトの分類をさらに微調整することができます：

- 新しい分類を 追加 したり、選択した分類を 削除 する、または分類を **すべて削除** することができます。
- クラス名の左にあるチェックボックスのチェックを外すことで、クラスを無効化することができます。無効化されたクラスに対応する地物はマップ上に表示されません。
- 行をドラッグ＆ドロップしてクラスの順序を並べ替えることができます。
- アイテムをダブルクリックすると、シンボルや値、クラスの凡例を変更することができます。

選択したアイテムを右クリックすると、コンテキストメニューが表示されます：

- **シンボルをコピーする** と **シンボルを貼り付ける** は、アイテムの表現を他のクラスに適用するのに便利な方法です。
- 選択したシンボルの **色を変更**...
- 選択したシンボルの **不透明度を変更**...
- 選択したシンボルの **出力単位を変更**...
- 選択したラインシンボルの **幅を変更する**...
- 選択したポイントシンボルの **サイズを変更**...
- 選択したポイントシンボルの **角度を変更**...
- **カテゴリをマージ** ：選択した複数のカテゴリを 1 つにまとめます。これにより、多数のカテゴリを持つレイヤのスタイル設定を簡単にできます。多数の相異なるカテゴリを、複数の値に適用される少数のより管理しやすいカテゴリの組にまとめることが可能になります。

  Tip: マージされたカテゴリのシンボルは、リスト内で選択されたカテゴリのうち最上位のものとなるため、マージする前に再利用したいシンボルのカテゴリを上に移動させると良いでしょう。

- 以前にマージされた **カテゴリをアンマージ**

作成したルールは **レイヤ** パネルのツリー階層にも表示されます。マップの凡例でエントリをダブルクリックすると、適用されたシンボルを編集できます。右クリックすると **さらにオプション** が表示されます。

詳細設定 メニューのオプションにアクセスして、分類を効率化したり、シンボルレンダリングを微調整することができます。

- **保存されたシンボルに一致** ： **シンボルのライブラリ** を使用して、分類のクラス値と名前が一致するシンボルを各カテゴリに割り当てます。
- **ファイルからのシンボルに一致**... ：シンボルが保存されたファイルを指定して、分類のクラス値と名前が一致するシンボルを各カテゴリに割り当てます。
- **描画順序**... は、シンボルのレンダリング順序を定義できます。

連続値レンダラー

 **連続値による定義（*graduated*）** レンダラーは、選択した地物の属性値のクラス割り当てを反映した色やサイズのユーザー定義のシンボルを使用して、レイヤの地物すべてをレンダリングするために使用します。

カテゴリ値レンダラーと同じように、連続値レンダラーでは特定の列について回転やサイズ縮尺の値を定義できます。

また、カテゴリ値レンダラーと同じように、以下の選択ができます：

- 分類の **値** ： これには既存のフィールドまたは **式** が使用できます。式はボックスに入力するか、関連する  ボタンを使用して構築することができます。 連続値に式を使用することで、シンボロジのためにフィールドを作成する必要がなくなります（例えば、分類基準が複数の属性値に由来する場合など）。

- シンボル（シンボルセレクタダイアログを使用します）
- 凡例のフォーマットと精度
- シンボルの変化に使用する方法：色または大きさ
- 色（カラーランプのリストを使用）：方法で色を選択した場合
- 大きさ（大きさの定義域およびその単位）

それから、割り当てられたフィールドまたは式の値のインタラクティブなヒストグラムを示す「ヒストグラム」タブを使用できます。ヒストグラムウィジェットを使用して、クラス分割の移動や追加ができます。

注釈: ベクタレイヤに関する詳細な情報を取得するために、統計パネルを使用できます。 **統計量の出力パネル** を参照してください。

「クラス」タブに戻って、クラスの数および地物をクラスに分類するためのモードも（「モード」リストを使用して）指定できます。使用可能なモードは以下のとおりです：

- 等量分類（Quantile）：各クラスは同じ数の要素を持ちます（ボックスプロットの考え方）。
- 等間隔分類：各クラスは同じサイズを持ちます（例：値の範囲が 1 から 16 で、4 つのクラスとすると、各クラスのサイズは 4 です）
- 対数スケール：広い範囲の値を持つデータに対して適しています。小さい値は狭いクラス、大きな値は広いクラスになります（例：範囲 [0..100] の 10 進数で、2 つのクラスとすると、1 つ目のクラスは 0 から 10 まで、2 つ目のクラスは 10 から 100 までとなります）。
- 自然分類（Jenks）：各クラス内の分散を最小化し、クラス間の分散を最大化するように分類します。
- 丸め間隔（Pretty）：x の値の範囲をカバーする、約 n+1 個の等間隔のちょうどよい値の列を計算します。値は 10 のべき乗の 1、2 または 5 倍になるように選択されます（R 言語の pretty https://www.rdocumentation.org/packages/base/topics/pretty に基づいています）。
- 標準偏差：クラスは値の標準偏差に応じて形成されます。

シンボロジ タブの中央にあるリストボックスには、クラスと値の範囲、凡例、レンダリングされるシンボルがリストされます。

分類 ボタンをクリックすると、選択したモードを使用してクラスが作成されます。各クラスは、クラス名の左にあるチェックボックスのチェックを外すことで無効にできます。

シンボルや値、クラスの凡例を変更するには、変更したい項目をダブルクリックします。

選択したアイテムを右クリックすると、コンテキストメニューが表示されます：

- **シンボルをコピーする** と **シンボルを貼り付ける** は、アイテムの表現を他のクラスに適用するのに便利な方法です。
- 選択したシンボルの **色を変更**...
- 選択したシンボルの **不透明度を変更**...
- 選択したシンボルの **出力単位を変更**...
- 選択したラインシンボルの **幅を変更する**...

- 選択したポイントシンボルの **サイズを変更**...
- 選択したポイントシンボルの **角度を変更**...

図 16.5 の例は、QGIS サンプルデータセットの major_rivers レイヤに対する連続値レンダリングダイアログを示しています。

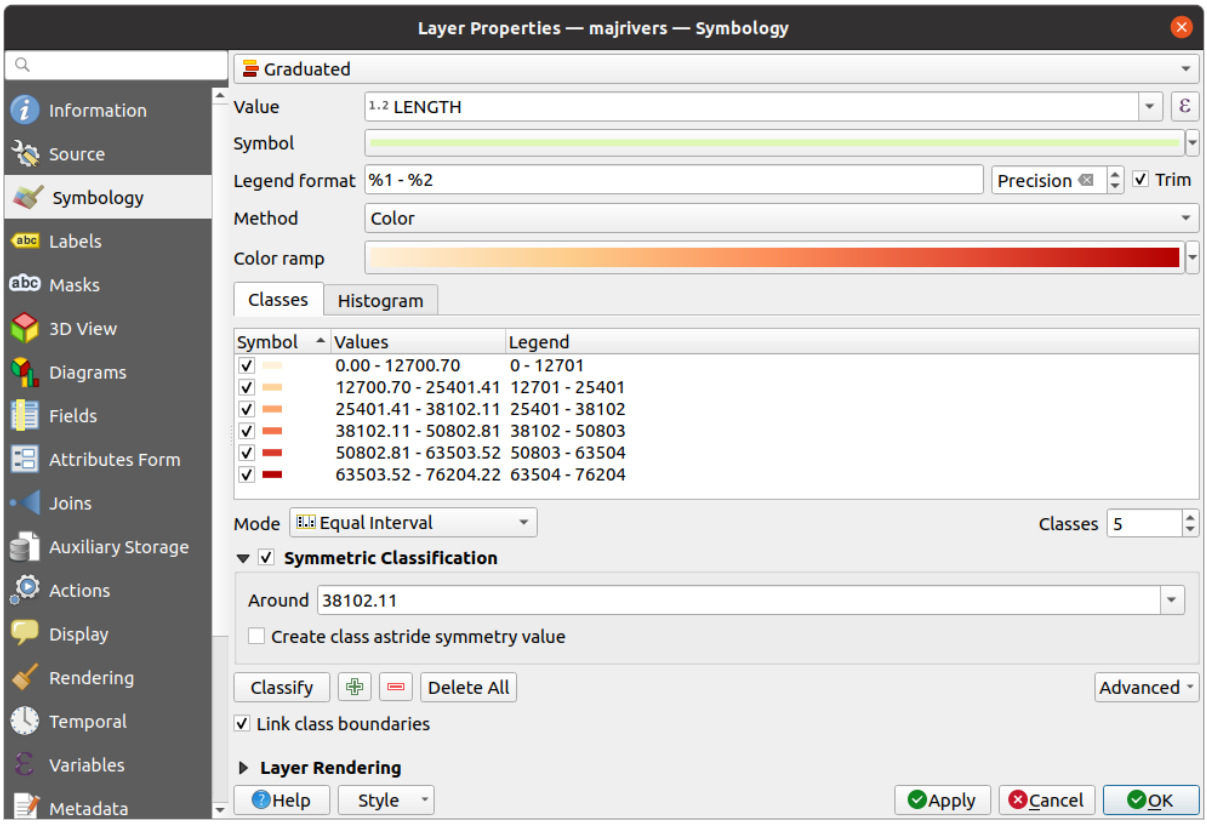


図 16.5: 連続値シンボルのオプション

作成したルールは **レイヤ** パネルのツリー階層にも表示されます。マップの凡例でエントリをダブルクリックすると、適用されたシンボルを編集できます。右クリックすると **さらにオプション** が表示されます。

比例シンボルと多変量解析

比例シンボルや多変量解析は、シンボロジのレンダリングドロップダウンリストから利用可能なレンダリングタイプではありません。しかし、上に挙げた任意のレンダリングオプションと **データによって定義された上書き** オプションを組み合わせることで、QGIS においてポイントデータやラインデータをそのような表現で表示することができます。

比例シンボルを作成する

比例シンボルレンダリングを適用するには、以下の手順で操作します：

1. まずは、レイヤに **単一シンボルレンダラー** を適用します。
2. 次に、地物に適用するシンボルを設定します。

3. シンボルツリーの上位のアイテムを選択し、**大きさ**（ポイントレイヤ）や **幅**（ラインレイヤ）オプションの隣にある  データによって定義された上書き **ボタン** を使用します。

4. フィールドを選択するか式を入力すると、各地物について QGIS は出力値をプロパティに適用し、マップキャンバスのシンボルを比例したサイズに変更します。

   必要ならば、 メニューの **アシスタント**... オプションを使用することで、シンボルサイズに何らかの変換（指数関数的、フラナリー...）を適用し、サイズの再スケーリングができます（詳細は **データによる定義のアシスタントインターフェースを使用する** 参照）。

レイヤパネル 内や **印刷レイアウトの凡例アイテム** で比例シンボルを表示するかどうかを選択することができます。**シンボロジ** タブのメインダイアログの下部にある **詳細設定** ドロップダウンリストを開き、 **データによる凡例サイズの定義...** を選択して、凡例アイテムの設定ができます（詳細は **データ定義による凡例サイズ** を参照）。

多変量解析を作成する

多変量解析レンダリングを使用すると、複数の変数間の関係を評価するのに役立ちます。例えば、ある変数をカラーランプで表して、別の変数をサイズで表現することができます。

QGIS で多変量解析を作成する最も簡単な方法は、

1. はじめに、カテゴリ値または連続値によるレンダリングをレイヤに適用します。すべてのクラスが同じタイプのシンボルです。

2. 次に、クラスに比例シンボロジを適用します：

   1. 分類フレームの上にある **変更** ボタンをクリックすると、 シンボルセレクタ ダイアログが開きます。

   2. 上で見たのと同様、 **データによって定義された上書き** ウィジェットを使用して、シンボルレイヤの大きさや幅を再スケーリングします。

比例シンボルと同様に、カテゴリ値あるいは連続値クラスシンボルの上には、 **データによる凡例サイズの定義** 機能を使用してスケーリングされたシンボロジをレイヤツリーに追加することができます。また、これらの表現は印刷レイアウトの凡例アイテムでも利用可能です。

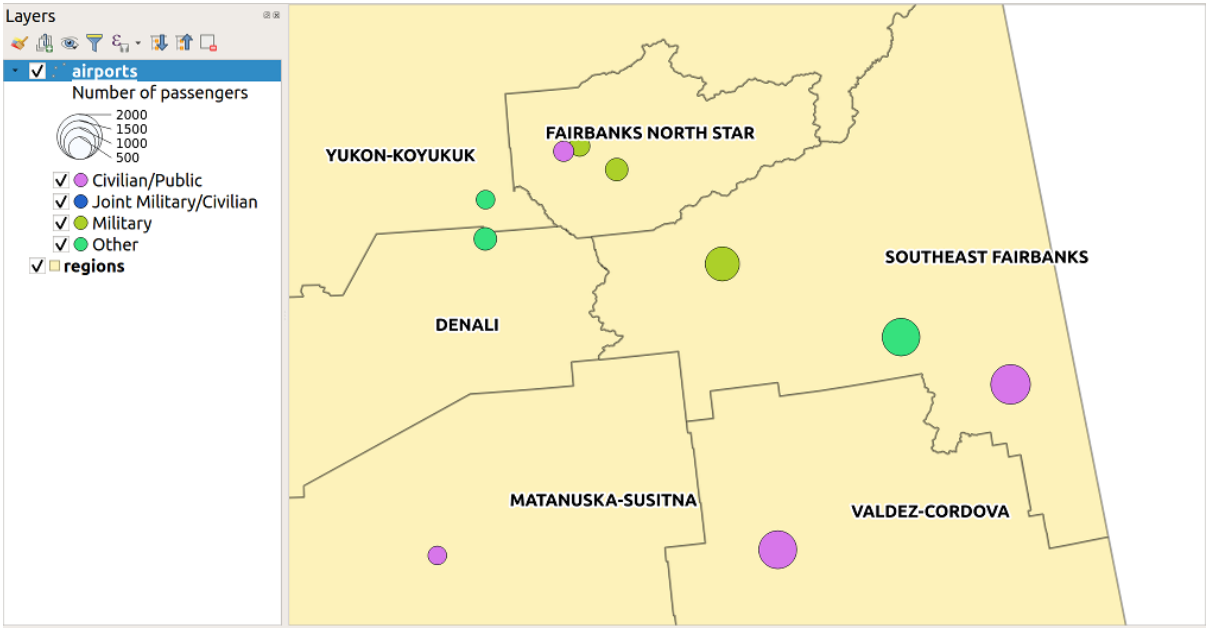


図 16.6: 凡例サイズを変化させた多変量解析の例

ルールベースレンダラー

ルールは QGIS の **式** で、属性やプロパティによって地物を区別し、特定のレンダリング設定をその地物へ適用するために使用されます。ルールは入れ子にすることが可能で、地物は入れ子になった上位のレベルすべてに属している場合に、あるクラスに属します。

ルールによる定義（*rule-based*） レンダラーは、選択した地物を細分化されたクラスに割り当てることを反映する特性を持ったシンボルを使用してレイヤの地物すべてをレンダリングするために設計されています。

ルールを作成するには：

1. 既存の行をダブルクリックしてアクティブにするか（デフォルトでは、レンダリングモードが有効な場合、QGIS はルールなしのシンボルを追加します）、または 現在のルールを編集 ボタンや ルールを追加 ボタンをクリックします。
2. 開いた **ルールの編集** ダイアログにおいて、各ルールを識別しやすいようにラベルを定義することができます。このラベルは **レイヤパネル** 内や印刷レイアウトの凡例にもに表示されます。
3. **フィルタ** オプションの隣にあるテキストボックスに式を手入力するか、テキストボックスの隣にある ε ボタンを押し、式文字列ビルダダイアログを開きます。
4. 提供されている関数とレイヤ属性を使用して、検索したい地物をフィルタリングするための **式** を作成します。 **テスト** ボタンを押すと、クエリの結果が確認できます。
5. ルールの詳しい説明のため、より長いラベルを入力することができます。
6. **縮尺の範囲** オプションを使用して、ルールが適用される縮尺を設定することができます。
7. 最後に、地物に **使用するシンボル** を設定します。

8. そして、 *OK* ボタンを押します。

ルールを要約した新しい行がレイヤプロパティダイアログに追加されます。上記の手順に従いルールを作成するか、既存のルールをコピー＆ペーストして、必要なだけルールを作成できます。ルールをドラッグ＆ドロップで上に重ねることで、ルールを入れ子にし、上位のルールの地物をサブクラスで改良できます。

ルールによる定義レンダラーは、カテゴリ値や連続値によるレンダラーと組み合わせることができます。ルールを選択し、 **選択したルールの絞り込み** ドロップダウンメニューを使用して、その機能をサブクラスでまとめることができます。絞り込みされたクラスはルールのサブアイテムのようにツリー階層で表示され、その親と同様に、各クラスでシンボロジやルールを設定できます。ルールの自動的な再設定は、以下の要素に基づいて行われます：

- **スケール** ：指定された縮尺のリストに基づいて、このオプションはさまざまなユーザー定義の縮尺範囲を適用する一連のクラスを作成します。スケールに基づいた新たな各クラスは、独自のシンボロジや式の定義を持つことができます。これは例えば、さまざまな縮尺で同じ地物を異なるシンボルで表示したり、縮尺に応じて一部の地物だけを表示するのに便利な方法です（例えば、ローカル空港は大きなスケールで、国際空港は小さなスケールで表示するなど）。
- **カテゴリ** ：選択したルールに合致する地物に **カテゴリ値レンダラー** を適用します。
- **範囲** ：選択したルールに合致する地物に **連続値レンダラー** を適用します。

絞り込まれたクラスはルールのサブアイテムのようにツリー階層で表示され、上と同様に各クラスにシンボロジを設定できます。入れ子になったルールのシンボルは上に積み重なっていくため、その選択には注意が必要です。 **ルールの編集** ダイアログで **シンボル** のチェックを外すことで、特定のシンボルをスタック内でレンダリングしないようにできます。

ルールの編集 ダイアログでは、全てのルールは記述せず、 **もしくは** オプションを使用して、同じレベルの他のどんなルールにもマッチしない地物全てをキャッチすることができます。これは、 **レイヤプロパティ　シンボロジ　ルールによる定義** ダイアログの **ルール** 列に `Else` と入力することでも実現できます。

選択したアイテムを右クリックすると、コンテキストメニューが表示されます：

- **コピー** と **貼り付け** ：既存のアイテムをもとに新しいアイテムを作成するのに便利な方法です。
- **シンボルをコピーする** と **シンボルを貼り付ける** は、アイテムの表現を他のクラスに適用するのに便利な方法です。
- 選択したシンボルの **色を変更**...
- 選択したシンボルの **不透明度を変更**...
- 選択したシンボルの **出力単位を変更**...
- 選択したラインシンボルの **幅を変更する**...
- 選択したポイントシンボルの **サイズを変更**...
- 選択したポイントシンボルの **角度を変更**...
- **現在のルールを再調整する** サブメニューを開き、現在のルールを **スケール** 、 **カテゴリ** または **範囲** で絞りこむことができます。ダイアログ下部にある **対応したメニュー** を選択したときと同様です。

ルールによる定義レンダラーダイアログの行のチェックを外すと、特定のルールやその入れ子になったルールに当てはまる地物がマップキャンバスから隠されます。

作成したルールはマップの凡例においてもツリー階層で表示されます。マップの凡例でエントリをダブルクリックすると、適用されたシンボルを編集できます。右クリックすると **さらにオプション** が表示されます。

図 16.7 の例は、QGIS サンプルデータセットの河川レイヤに対するルールによるレンダリングダイアログを示しています。

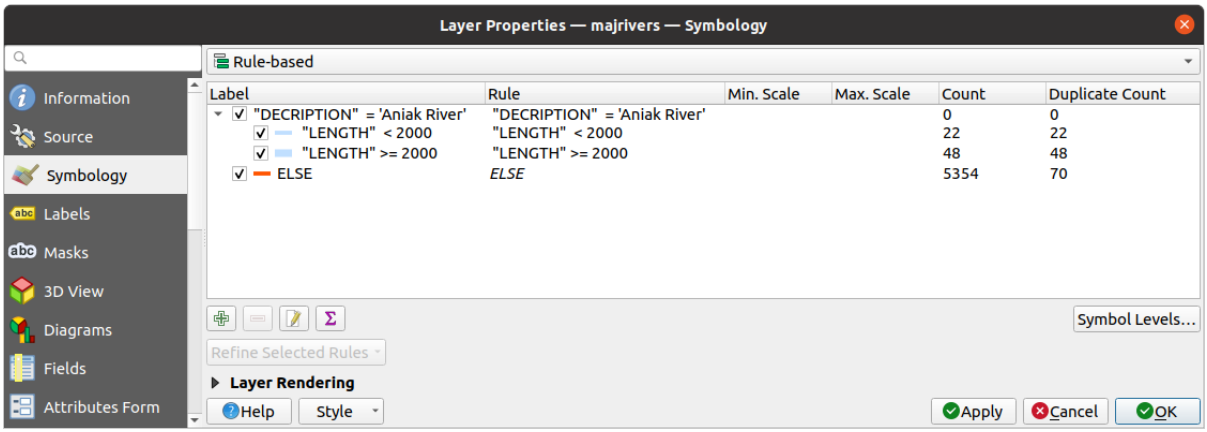


図 16.7: ルールによる定義シンボルのオプション

点の競合回避レンダラ

点の競合回避 レンダラは、互いに与えられた距離の許容範囲内にあるポイント地物を取り、異なる配置方法に従って、それらのシンボルをそれらの重心の周りに配置します。これは、点レイヤの地物が同じ場所にあっても（例 建物内の施設）、すべての地物を視覚化する便利な方法です。

点の競合回避レンダラを設定するには:

1. **中央シンボル** を設定します: 中心にある仮想的な点がどのように表示されるか
2. **レンダラ** の種類を選びます: レイヤ内の地物をどのように分類するか（単一、カテゴリ値、ルール...）
3. **レンダラの設定**... ボタンを押して選択したレンダラに応じた地物のシンボロジを設定します
4. 近い地物が重なっているとみなされ、同じ仮想的な点上で競合回避する **距離** 許容範囲を設定します。共通のシンボル単位をサポートします。
5. **競合する点の配置方法** を設定します:
   - **リング状に配置** ：すべての地物を一つの円周上に配置します。円周の半径は表示する地物の数に依存します。
   - **同心円状に配置** ：地物を表示するために同心円の集合を使用します。
   - **グリッド** ：規則的な格子を生成し、格子の交点にポイントシンボルを配置します。
6. 競合回避シンボルは **置き換え線** 上に配置されます。置き換え線の最小間隔はポイントシンボルレンダラに依存しますが、**ストローク幅**、**ストローク色**、**サイズ調整** などの設定をカスタマイズすることができます（例えば、レンダリングされたポイントの間隔を広げるなど）。

7. **ラベル** グループオプションを使用して、ポイントのラベル付けを行います: ラベルは競合回避シンボルの近くに配置され、地物の実際の位置には配置されません。

    1. **ラベル属性**: ラベル付けに使うレイヤのフィールドを選択します。
    2. **ラベルフォント** プロパティと大きさを設定します
    3. **ラベル色** を選びます
    4. **ラベルの距離係数** を設定します: 各ポイント地物のラベルを、シンボルの対角サイズに比例して、シンボルの中心からオフセットします。
    5. ☐ **縮尺依存ラベルを使用する** をオンにすると、指定した **地図の最小縮尺** より大きな縮尺のときだけラベルを表示します。

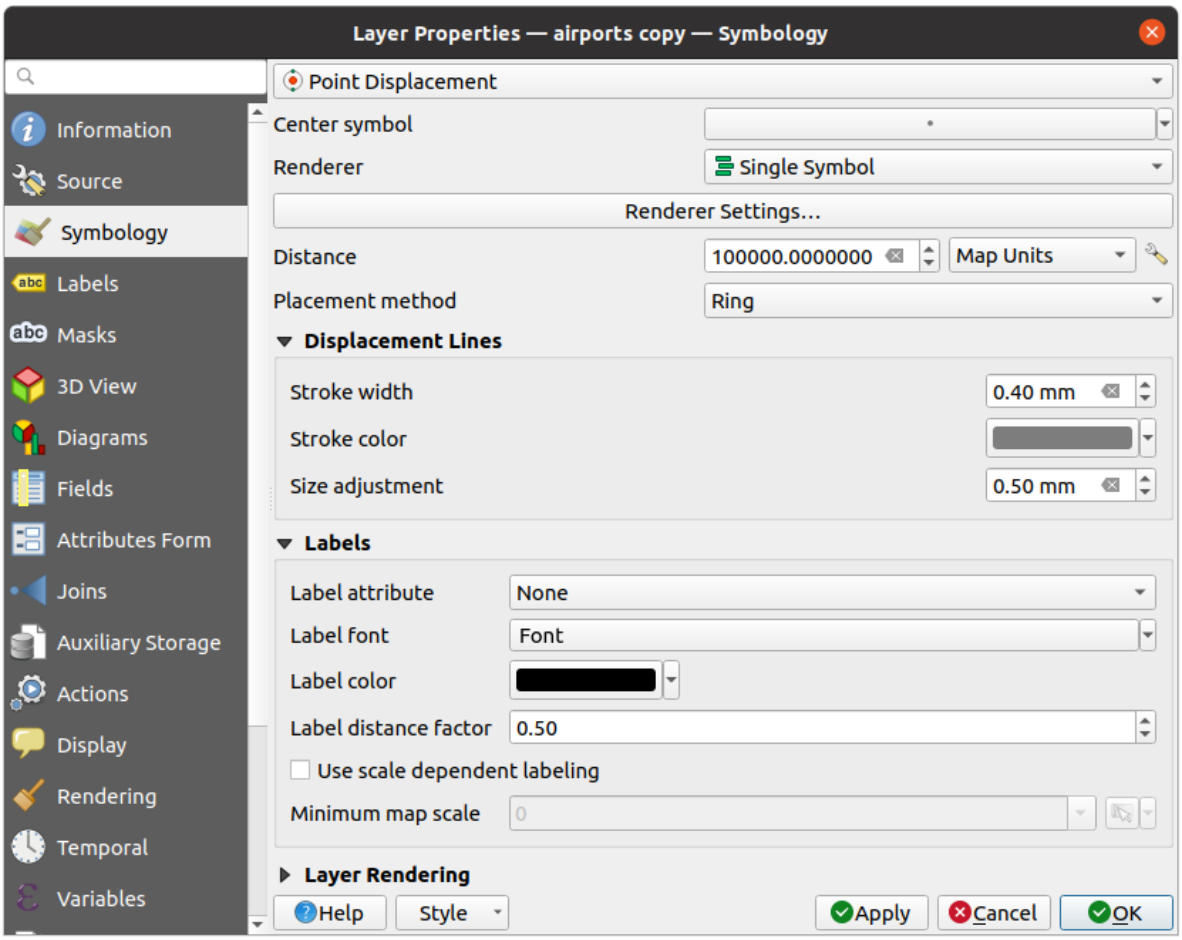


図 16.8: 点の競合回避ダイアログ

注釈: 点の競合回避レンダラーは、地物のジオメトリは変更しません。つまり、ポイントはその位置から移動してはいません。ポイントは元の場所に位置しています。変更はレンダリング目的のための視覚的なものに過ぎません。ずらした地物を作成したい場合には、代わりにプロセシングアルゴリズムの **近隣点を散らす** を使用してください。

点のクラスタレンダラー

点の競合回避 レンダラーが最も近いか重ね合うポイント地物の配置を吹き飛ばすのとは異なり、**点のクラスタ** レンダラーは近接した点を 1 つのレンダリングされたマーカーシンボルにグループ化します。互いに指定された距離内にある点は 1 つのシンボルに統合されます。点の集約は、単に検索距離内の最初のグループを割り当てるのではなく、最も近いグループが形成されるように行われます。

メインダイアログでは、以下の設定ができます。

1. **クラスタシンボル** に点のクラスタを表すシンボルを設定する; デフォルトのレンダリングでは、フォントマーカーシンボルレイヤの `@cluster_size` *variable* により集約された地物の数が表示されます。
2. **レンダラ** の種類、つまりレイヤ内の地物をどのように分類したいか（単一、カテゴリ値、ルール...）を選びます
3. **レンダラの設定** ボタンを押すと、地物のシンボルを通常通り設定することができます。このシンボルはクラスタ化されていない地物にのみ表示され、それ以外の地物には **クラスタシンボル** が適用されることに注意してください。また、クラスタ内のすべての点地物が同じレンダリングクラスに属し、同じ色が適用される場合、その色はクラスタの `@cluster_color` 変数を表します。
4. 地物をクラスタ化する際に考慮する **距離** の最大値を指定します。共通のシンボル単位をサポートします。

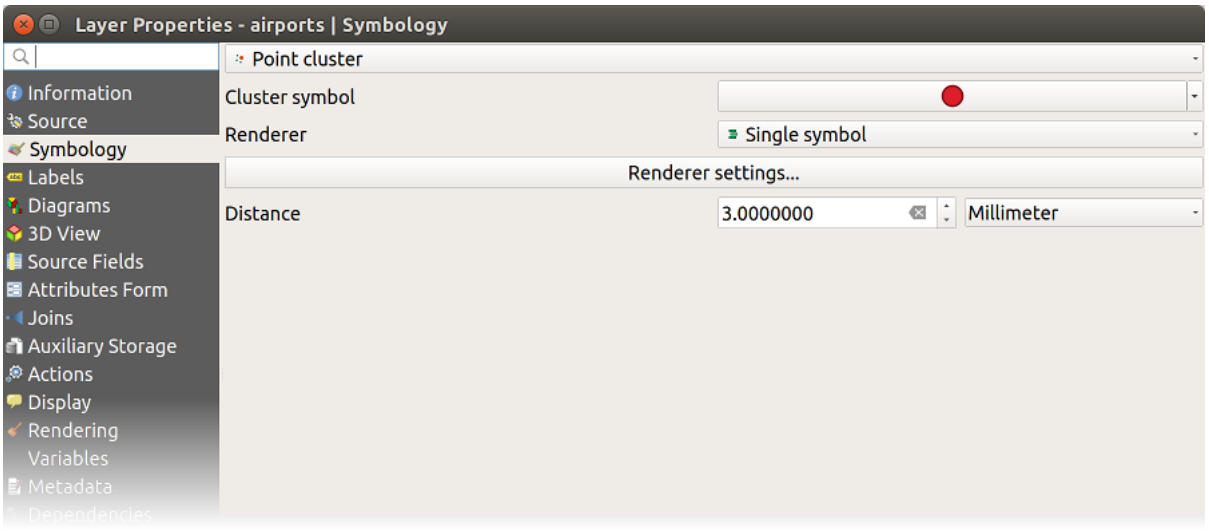


図 16.9: 点のクラスタダイアログ

注釈: 点のクラスタレンダラーは、地物のジオメトリは変更しません。つまり、ポイントはその位置から移動してはいません。ポイントは元の場所に位置しています。変更はレンダリング目的のための視覚的なものに過ぎません。クラスターに基づく地物を作成したい場合には、代わりにプロセシングアルゴリズムの *K* **平均クラスタリング** または *DBSCAN* **クラスタリング** を使用してください。

結合済み地物レンダラー

結合済み地物 レンダラーは、レンダリング前にポリゴン地物やライン地物を単一のオブジェクトに“融合させる”ことができます。これにより、複雑なシンボルや重なった地物が単一の連続的な地図シンボルとして表現されます。

反転ポリゴンレンダラー

反転ポリゴン レンダラーでは、レイヤのポリゴンの外側を塗りつぶすシンボルを定義できます。上記と同様、サブレンダラー、すなわち単一シンボル、連続値、カテゴリ値、ルールに基づく、または 2.5D レンダラーを選択できます。

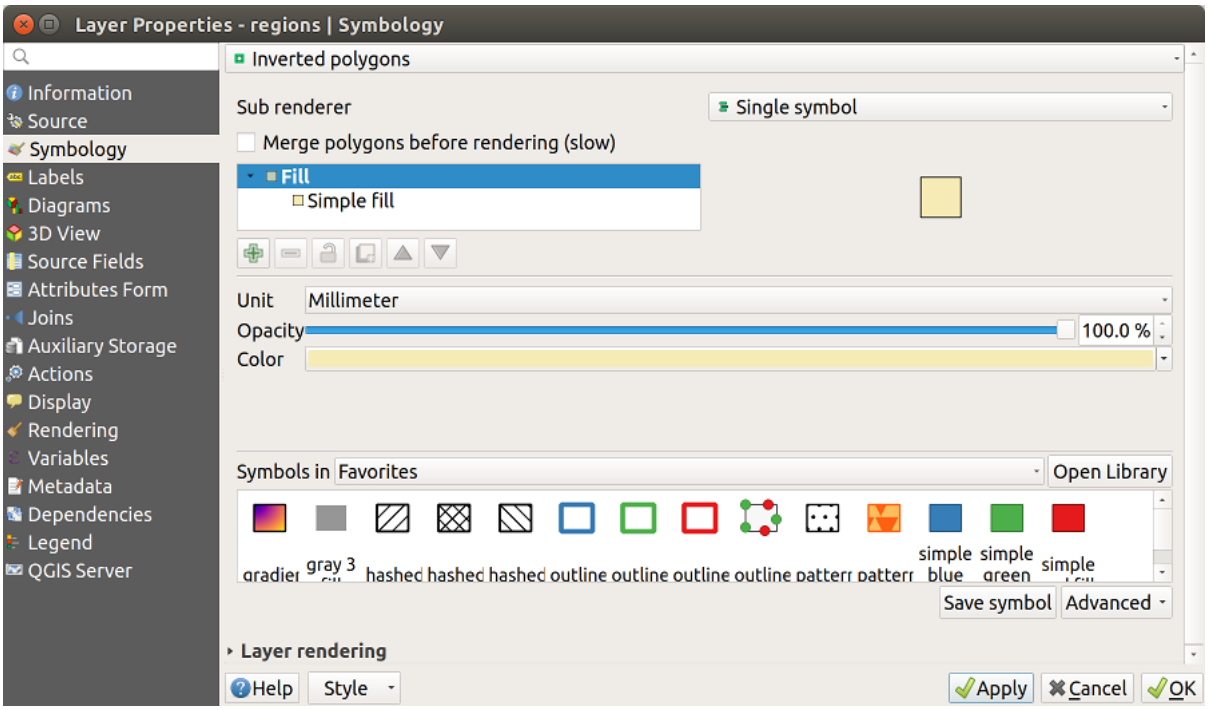


図 16.10: 反転ポリゴンダイアログ

ヒートマップレンダラー

ヒートマップ レンダラーを使用すると、（マルチ）ポイントレイヤ用のライブダイナミックヒートマップを作成できます。ヒートマップ半径をミリメートル、ポイント、ピクセル、地図単位、インチで指定したり、ヒートマップのスタイルのカラーランプの選択・編集をしたり、スライダーを使用してレンダリングの速度と品質のトレードオフを選択したりすることができます。また、最大値の上限を定義したり、フィールドや式を使用してポイントに重みづけを与えることもできます。地物を追加したり削除したりすると、ヒートマップレンダラは自動的にヒートマップスタイルを更新します。

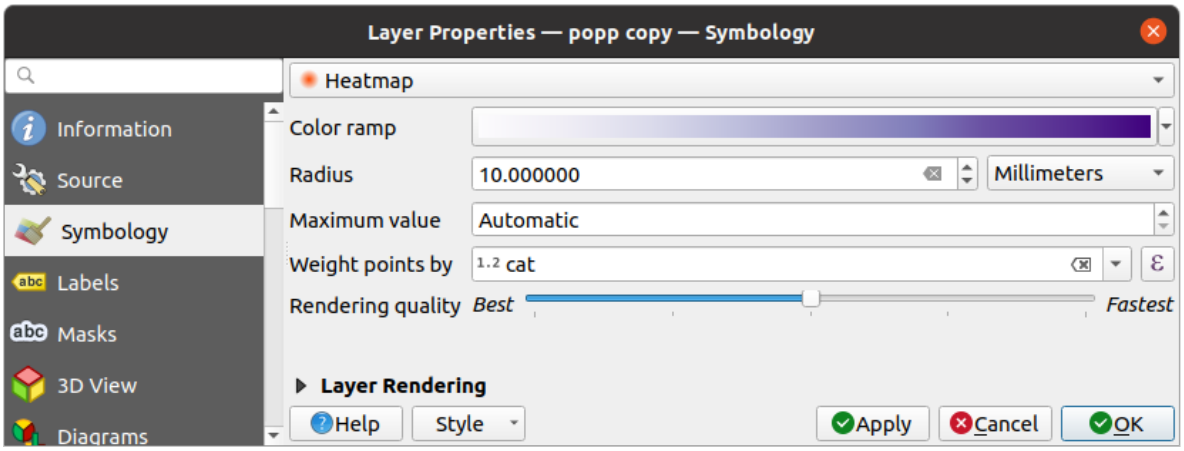


図 16.11: ヒートマップダイアログ

2.5D レンダラー

2.5D レンダラーを使用すると、レイヤの地物に対して 2.5 次元的な効果を作成することができます。まずは、 **高さ** の値を（マップ単位で）選択します。これには、固定値、レイヤのフィールドのいずれか、または式を使用することができます。また、視点位置を再現するために **角度** を（度単位）を選択する必要があります（0 ° は西を意味し、反時計回り方向に数字が大きくなります）。詳細設定オプションで、 **屋根の色** と **壁の色** を設定します。地物の壁に太陽放射をシミュレートしたい場合には、 **壁の向きに応じた陰影表現** オプションにチェックを入れてください。また、影の **色** と **大きさ** （マップ単位）を設定することで、影をシミュレートすることもできます。

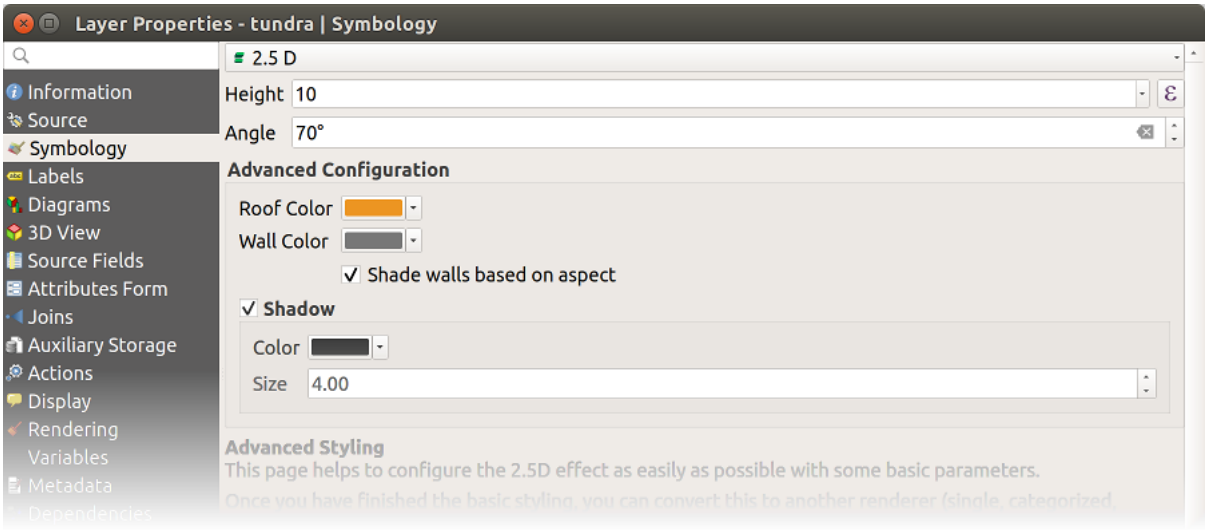


図 16.12: 2.5D ダイアログ

Tip: 2.5D 効果を他のレンダラーと併用する

2.5D レンダラーの基本的なスタイルの設定が完了したら、これを別のレンダラー（単一、カテゴリ値、連続値）に変換できます。2.5D 効果は保持され、他のすべてのレンダラー固有のオプションを使用して調整

することができます（これにより、例えばカテゴリシンボルに素敵な 2.5 次元表現を加えたり、2.5D シンボルにさまざまな追加のスタイリングを加えられます)。影や「建物」自体が他の近くの地物と干渉しないようにするためには、描画順序を有効にする必要がある場合があります（ **詳細設定** **描画順序**...)。2.5D の高さと角度の値はレイヤ内の変数に保存されているので、後からレイヤのプロパティダイアログボックスの変数タブで編集することができます。

埋め込みシンボルレンダラー

埋め込みシンボル レンダラーは、データソースが提供する「ネイティブな」シンボルを表示します。これは主に、既定のシンボロジを持つ KML や TAB データセットで利用します。

レイヤレンダリング

シンボロジタブでは、レイヤの全ての地物に共通で作用するいくつかのオプションを設定することもできます：

- **不透明度**  ：このツールを使用すると、マップキャンバスで背面にあるレイヤを見えるようにできます。スライダーを使用して、ベクタレイヤの見え方を必要に応じて変化させてください。スライダーの横にあるメニューで不透明度の割合を正確に定義することもできます。
- **レイヤ** や **地物** レベルの **混合モード** ：このツールを使用すると、これまではグラフィックソフトでしか使えなかったような、特別なレンダリング効果が得られます。上下のレイヤのピクセルは、 **混合モード** で説明されている設定で混合されます。
- **描画エフェクト** ボタンを使用して、レイヤの地物すべてに **描画効果** を適用します。
- **地物描画順序の制御** では、地物属性を使用して、地物のレンダリングの Z-順序 を定義できます。チェックボックスを有効にし、横にある  ボタンを押してください。 **順序を定義する** ダイアログが開き、以下の手順で描画順序の定義ができます：

  1. フィールドを選択するか、レイヤの地物に適用する式を作成します。
  2. 取得した地物をどの順序で並べるかを設定します。つまり、 **昇順** を選択した場合には、低い値の地物が高い値の地物よりも下にレンダリングされます。
  3. 地物が NULL 値を返す時のレンダリングについて定義します： **NULL は最初に** （一番下に）または **NULL は最後に** （一番上に）
  4. 使用したいルールの数だけ上記の手順を繰り返します。

  最初のルールはレイヤ内の全ての地物に適用され、戻り値に応じて z-順序が付けられます。それから、同じ値（NULL 値も含む）を持った各グループ内の地物について、同じ z-順序を持っていることから、次のルールを適用してこれらを並べ替えます。以下、同様に続きます。

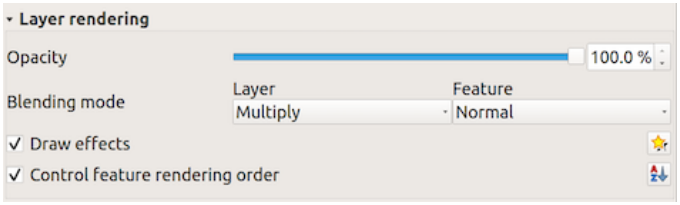


図 16.13: レイヤレンダリングオプション

その他の設定

シンボルレベル

重なったシンボルレイヤを許可するレンダラー（ヒートマップのみなし）のために、各シンボルレベルのレンダリング順序を制御するためのオプションがあります。

ほとんどのレンダラーについて、保存されたシンボルリストの下の **詳細設定** ボタンをクリックして **描画順序** を選択することで、シンボルレベルのオプションにアクセスすることができます。 **ルールベースレンダラー** では、このオプションは **描画順序**... ボタンから直接利用可能である一方で、 **点の競合回避レンダラ** レンダラーではこのボタンは **レンダラの設定** ダイアログ内にあります。

シンボルレベルを有効にするには、 **描画順序を有効にする** にチェックを入れます。各行には結合されたシンボルの小さなサンプルとそのラベル、個々のシンボルレイヤが表示され、個々のシンボルレイヤは横に番号が付いた列に分割されています。数字はシンボルレイヤが描画されるレンダリング順序のレベルを表します。低い値のレベルが最初に描画されて下に置かれ、高い値のレベルは最後に他のシンボルレイヤの上に描かれます。

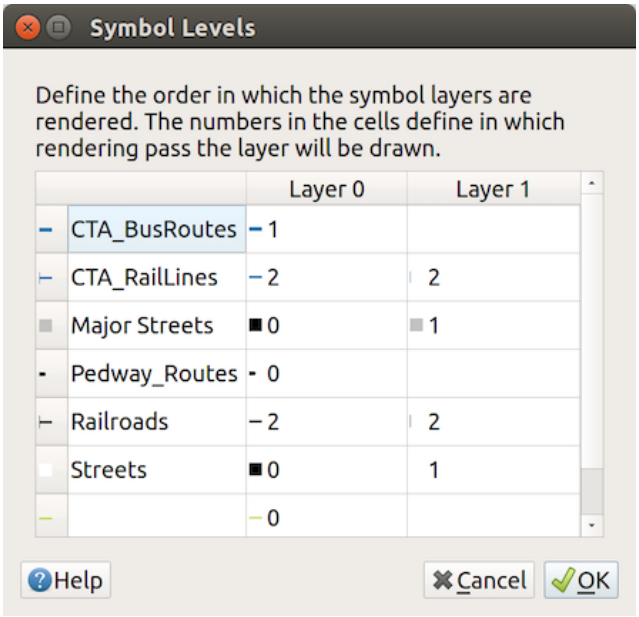


図 16.14: 描画順序ダイアログ

注釈: 描画順序を無効にしている場合には、完全なシンボルがそれぞれの地物の順序に従って描画されま

す。重なるシンボルは、他の下にあるシンボルを単に見えにくくします。加えて、同種のシンボルがお互いに「結合する」ことはありません。

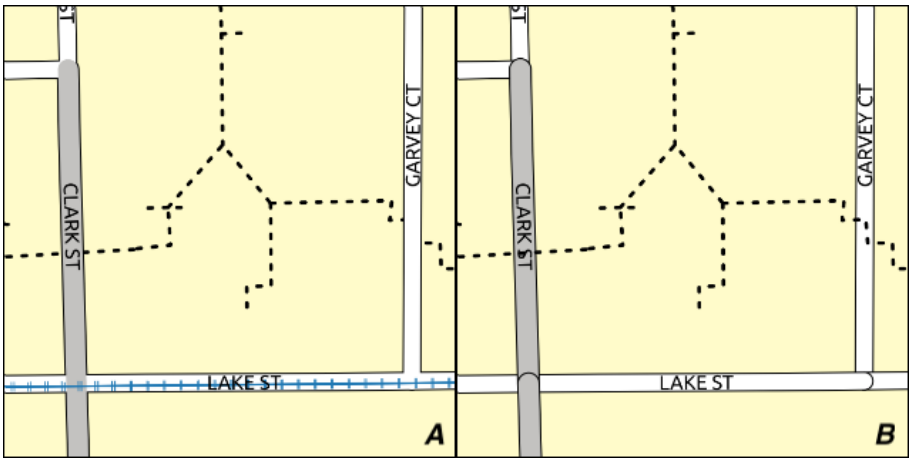


図 16.15: シンボルレベルの有効（A）と無効（B）の差

データ定義による凡例サイズ

レイヤが **比例シンボルや多変量解析レンダリング** によってレンダリングされている場合や、レイヤに **可変サイズダイアグラム** が適用されている場合には、 **レイヤパネル** や **印刷レイアウトの凡例** のどちらにもスケーリングされたシンボルを表示させることができます。

データによる凡例サイズの定義 ダイアログを有効にしてシンボロジをレンダリングするには、保存されたシンボルリストの下にある **詳細設定** ボタンの同名のオプションを選択します。ダイアグラムについては、このオプションは **凡例** タブにあります。このダイアログには、以下のオプションがあります：

- 凡例タイプの選択： **凡例が有効になっていません** 、 **リスト型凡例** 、 **重ね合わせ凡例** から選択します。重ね合わせ凡例オプションについては、凡例アイテムの整列を **下** か **中央** で選択できます。
- 凡例表現で使用する **シンボル** のプレビュー
- 凡例へのタイトルの挿入
- 使用するクラスのサイズの再設定：デフォルトでは、QGIS は（自然なプリティブレークに基づく）5 つのクラスの凡例を提供しますが、 **サイズ項目の指定** オプションを使用して独自の分類を適用することができます。 と ボタンを使用して、カスタムクラスの値とラベルを設定します。
- 重ね合わせ凡例の場合には、以下を設定できます：
  - **整列方法** ：下または中央
  - **ラインシンボル** の設定：シンボルから対応する凡例テキストへ引かれる水平線

凡例のプレビューがダイアログの右パネルに表示され、パラメータの設定に応じて更新されます。

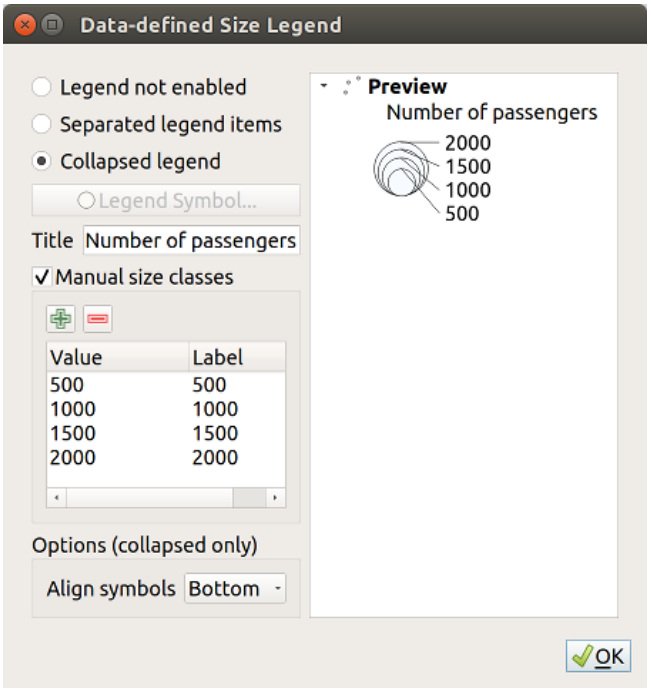


図 16.16: データによる凡例サイズの設定

注釈: 現在のところ、レイヤのシンボロジに対するデータによる凡例サイズの設定は、ポイントレイヤで単一定義、カテゴリ値、または連続値シンボロジを使用する場合にのみ適用できます。

描画エフェクト

レイヤのレンダリングを改善し、地図の最終レンダリングのために他のソフトウェアに頼ることを避ける（あるいは少なくとも減らす）ために、QGIS にはもう 1 つ強力な機能があります。 それは **描画エフェクト** オプションで、ベクタレイヤの可視化をカスタマイズするための描画効果を追加します。

このオプションは、 **レイヤプロパティ　　シンボロジ** ダイアログの **レイヤレンダリング** グループ（レイヤ全体に適用する場合）、または **シンボルレイヤプロパティ** （対応する地物に適用する場合）にて利用可能です。両方を組み合わせて使用することもできます。

描画効果は **描画エフェクト** オプションにチェックを入れ、 効果をカスタマイズ ボタンをクリックすることで有効化できます。これによって **効果のプロパティ** ダイアログ（ 図 16.17 参照）が開きます。以下のエフェクトタイプとカスタムオプションが利用可能です：

- **ソース** ：レイヤのプロパティの設定に応じて地物の元のスタイルを描画します。スタイルの **不透明度** や、 **混合モード** 、 **描画モード** の調整を行えます。これらはすべてのエフェクトタイプにに共通のプロパティです。

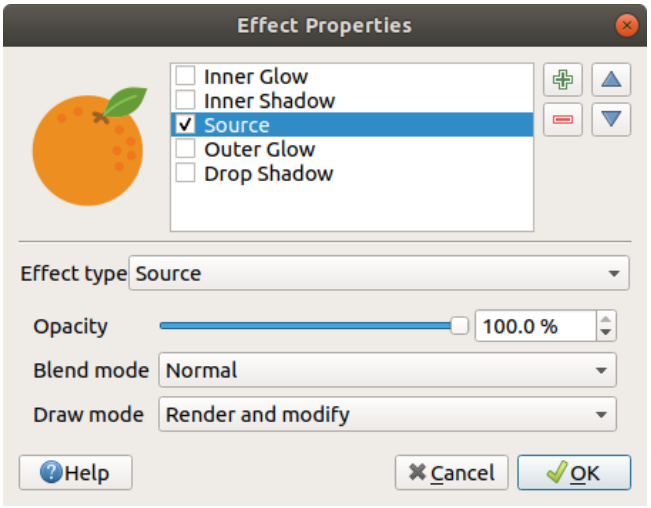


図 16.17: 描画エフェクト：ソースダイアログ

- **ぼかし** ：ベクタレイヤにぼかし効果を追加します。変更可能なカスタムオプションには、**ぼかしの種類** （ *Stack blur* **（高速・高 *dpi* 非対応）** または *Gaussian blur* **（高 *dpi* 対応）** ）と **ぼかしの強さ** があります。

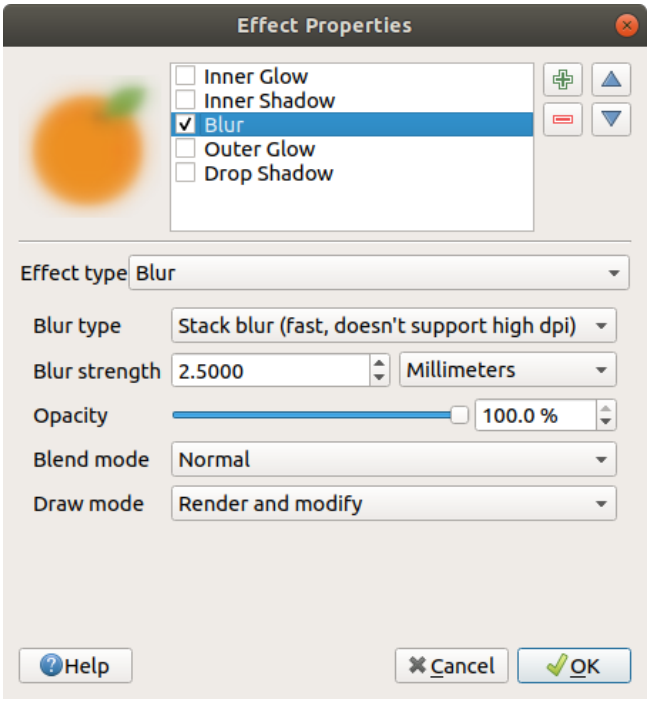


図 16.18: 描画エフェクト：ぼかしダイアログ

- **色付け** ：この効果は、単一の色相を使用したスタイルのバージョンを作成するために使用します。ベースは常にシンボルのグレースケールバージョンで、これに対し以下の設定ができます：
  - **グレースケール** は、グレースケール画像をどのように作成するかを選択するのに使用します：選択肢は、「明度を使用」、「光度を使用」、「平均を使用」、「オフ」です。
  - **着色** にチェックを入れると、別の色と混合することができ、色の強さを調整できます。

– 適用後のシンボルの **輝度** 、 **コントラスト** 、 **彩度** のレベルを調整できます。

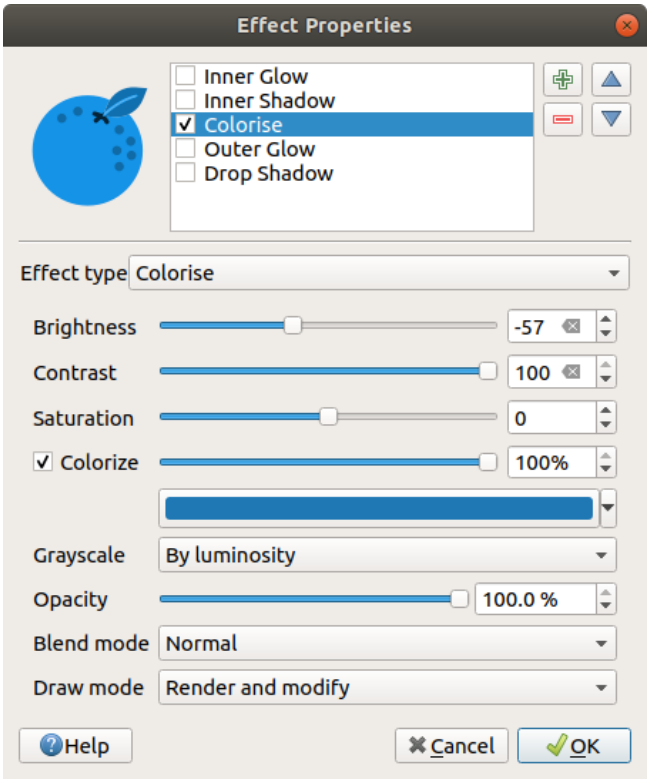


図 16.19: 描画エフェクト：色付けダイアログ

- **ドロップシャドウ** ：この効果を使用すると地物に影が追加され、奥行きが追加されたように見えます。この効果は、影が移動する方向とソースオブジェクトとの近さを決定する **オフセット** 角度・距離を変更することでカスタマイズできます。 **ドロップシャドウ** にはエフェクトの **ぼかし半径** と **色** を変更するオプションもあります。

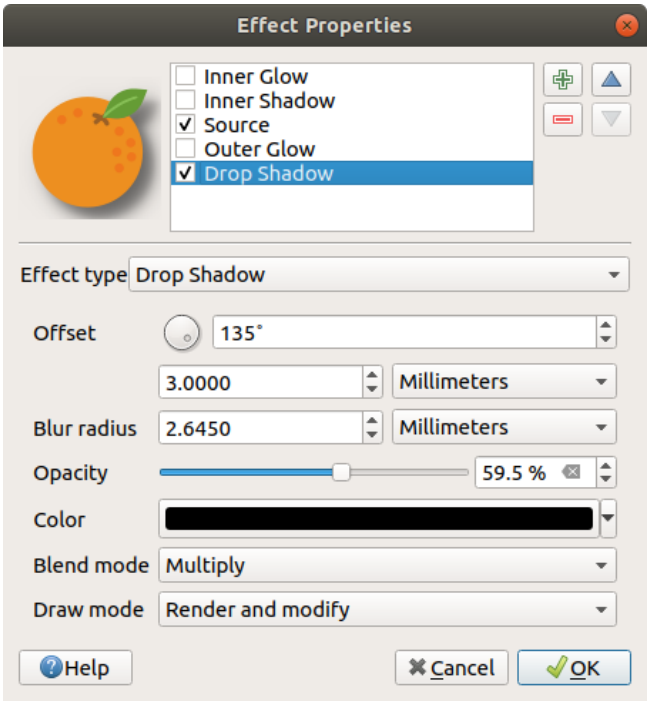


図 16.20: 描画エフェクト：ドロップシャドウダイアログ

- **インナーシャドウ** ：この効果は **ドロップシャドウ** 効果と似ていますが、こちらは地物の境界の内部に影効果を追加します。カスタマイズに関するオプションは **ドロップシャドウ** 効果のものと同様です。

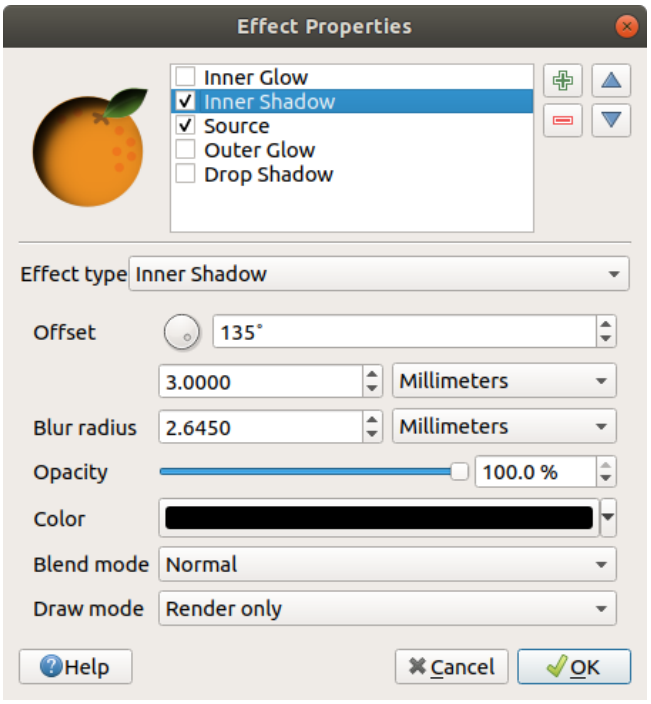


図 16.21: 描画エフェクト：インナーシャドウダイアログ

- **インナーグロー** ：地物内部にグロー効果を追加します。この効果は、グローの **広がり** （幅）や **ぼかし半径** を調整することでカスタマイズできます。後者は、ぼかしをかけたい地物の端からの距離を

指定します。加えて、 **単一色** または **カラーランプ** を使用してグローの色をカスタマイズするオプションもあります。

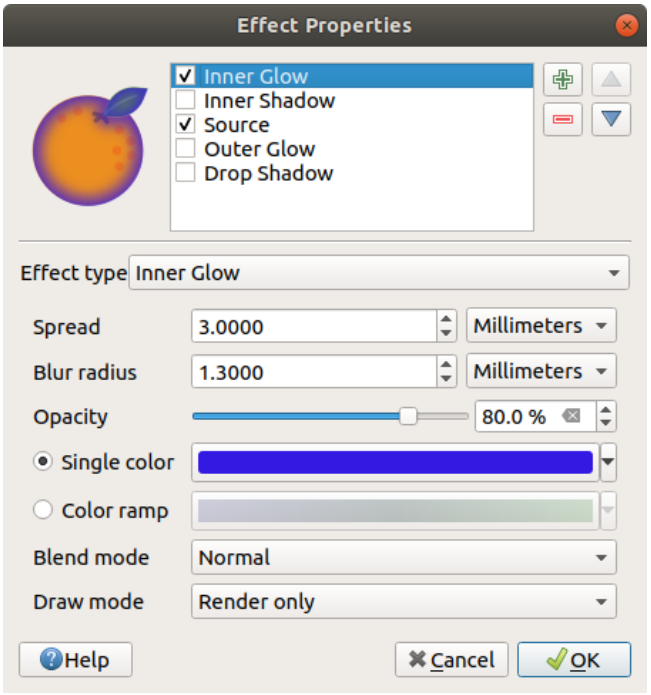


図 16.22: 描画エフェクト：インナーグローダイアログ

- **アウターグロー** ：この効果は **インナーグロー** 効果と似ていますが、こちらは地物の境界の外部にグロー効果を追加します。カスタマイズに関するオプションは **インナーグロー** 効果のものと同様です。

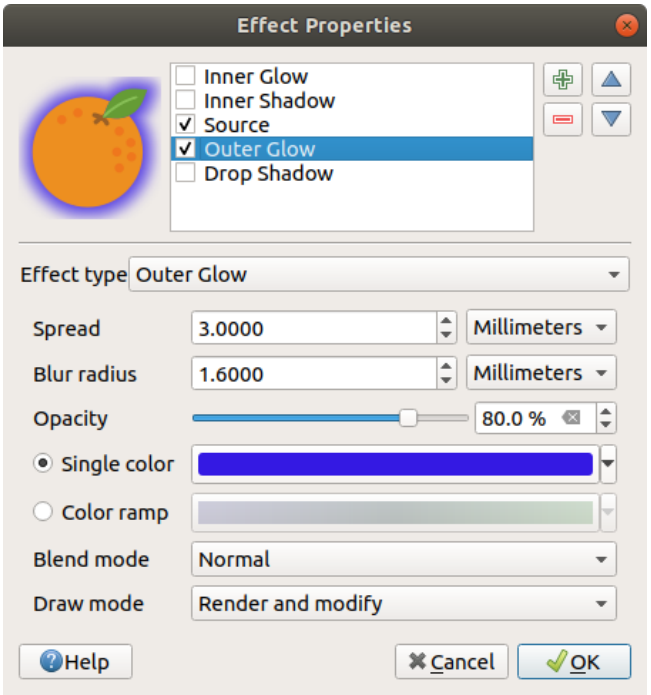


図 16.23: 描画エフェクト：アウターグローダイアログ

- **トランスフォーム** ：シンボル形状の変形可能性を追加します。カスタマイズに利用可能な 1 つ目のオプションは **水平反転** と **垂直反転** で、これらは実際にシンボルを水平・垂直軸に反転したものを作成します。その他のオプションは以下のとおりです：
  - **変形** *X,Y* ：地物を X 軸方向や Y 軸方向に歪めます。
  - **縮尺** *X,Y* ：X 軸や Y 軸に沿って与えられた割合で地物を拡大または縮小します。
  - **回転** ：地物を中心周りに回転させます。
  - **移動** *X,Y* ：与えられた X 軸や Y 軸方向距離に基づきアイテムの位置を変更します。

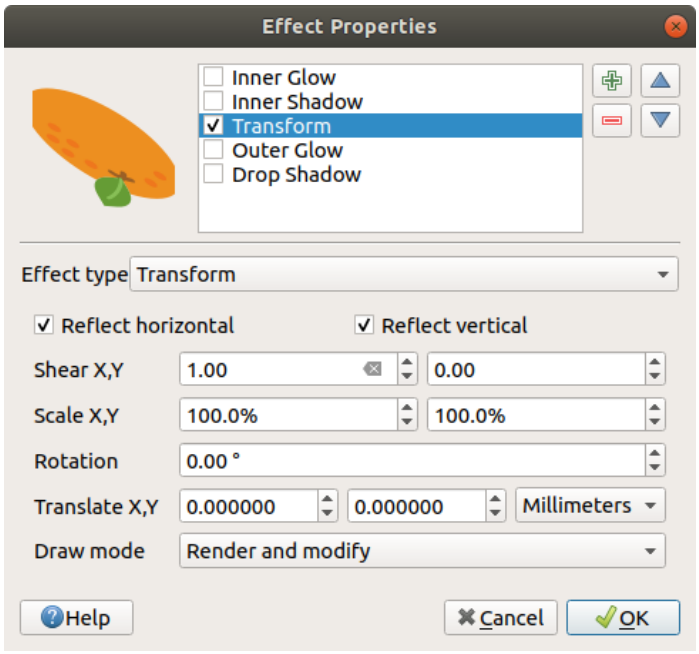


図 16.24: 描画エフェクト：トランスフォームダイアログ

複数のエフェクトタイプを同時に使用することができます。エフェクトリスト内のチェックボックスを使用して、エフェクトを有効化（無効化）できます。選択したエフェクトタイプを変更するには、 **エフェクトのタイプ** オプションを使用します。 上に移動 と 下に移動 ボタンを使用してエフェクトを並べ替えたり、 新しい効果を追加 や 効果を削除 ボタンを使用してエフェクトを追加/削除することもできます。

描画エフェクトで共通のオプションがあります。 **不透明度** と **混合モード** オプションは、 **レイヤレンダリング** での説明と同様に働き、トランスフォーム効果を除く全ての描画エフェクトで使用できます。

また、全てのエフェクトについて **描画モード** オプションが利用可能であり、下記のいくつかのルールに従い、シンボルのレンダリングや修正を行うかどうかについて選択できます。

- エフェクトは上から下の順でレンダリングされます。
- **レンダーのみ** モードは、エフェクトが見えることを意味します。
- **モディファイアのみ** モードは、エフェクトは見えないものの、適用した変更は次のエフェクト（すぐ 1 つ下のエフェクト）に渡されます。
- **レンダーとモディファイア** モードは、エフェクトが表示され、変更は次のエフェクトに渡されます。エフェクトがエフェクトリストの最上位またはすぐ上のエフェクトがモディファイアモードでない

場合には、レイヤプロパティのオリジナルのソースシンボルを使用します（「ソース」と同様です）。

16.1.4 ラベルプロパティ

ラベル プロパティは、ベクタレイヤに対するスマートなラベル付けを設定するために必要かつ適切なすべての機能を提供します。このダイアログには、 **レイヤスタイル** パネルや **ラベルツールバー** の レイヤラベリングオプション ボタンからもアクセスできます。

最初のステップは、ラベル付けの手法をドロップダウンリストから選択することです。利用可能な手法は、以下のとおりです：

- **なし** ：デフォルト値で、レイヤにラベルを表示しません。
- **単一定義** ：単一の属性もしくは式を使用して、マップにラベルを表示します。
- **ルールに基づく定義**
- **他レイヤのラベルをブロック** ：自分のラベルはレンダリングしませんが、レイヤが他のレイヤのラベルとの衝突を回避するように設定することができます。

単一定義 オプションを選択したとすると、次のステップは以下のダイアログが開きます。

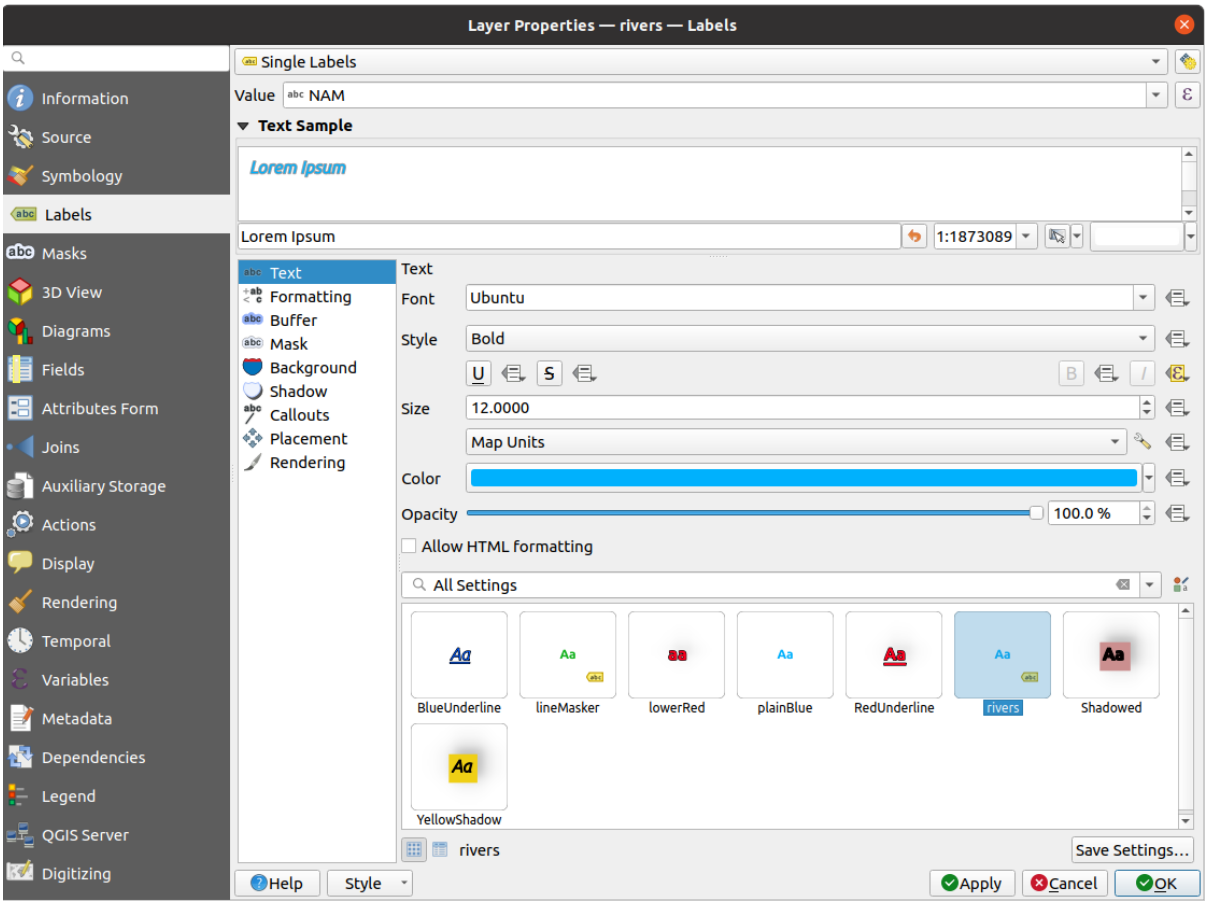


図 16.25: レイヤのラベル設定 - 単一定義

ダイアログの最上部には、 **値（*Value*）** のドロップダウンリストがあります。ラベル付けに使用する属性

カラムを選択することができます。デフォルトでは、 **表示名フィールド** が使用されます。式に基づいてラベルを定義したい場合には、 ε をクリックします。詳細は **式に基づいてラベルを定義する** を参照してください。

注釈: **凡例** タブでラベル凡例表示を有効にした場合、フォーマット付きのラベルが凡例内のエントリとして表示されます。

以下は、様々なタブの下でラベルをカスタマイズするために表示されるオプションです。

- abc **テキスト**
- **フォーマット**
- abc **バッファ**
- abc **マスクグリッド**
- **背景**
- **影**
- abc **引出し線付きラベル**
- **配置**
- **レンダリング**

各プロパティの設定方法は、 **ラベルの設定** にあります。

自動配置エンジンの設定

ラベルの自動的なふるまいに関するプロジェクトレベルの設定を行うために、自動配置設定を使用することができます。 **ラベル** タブの右上隅にある 自動配置設定（全レイヤに適用） ボタンをクリックするとダイアログが開き、以下の設定ができます：

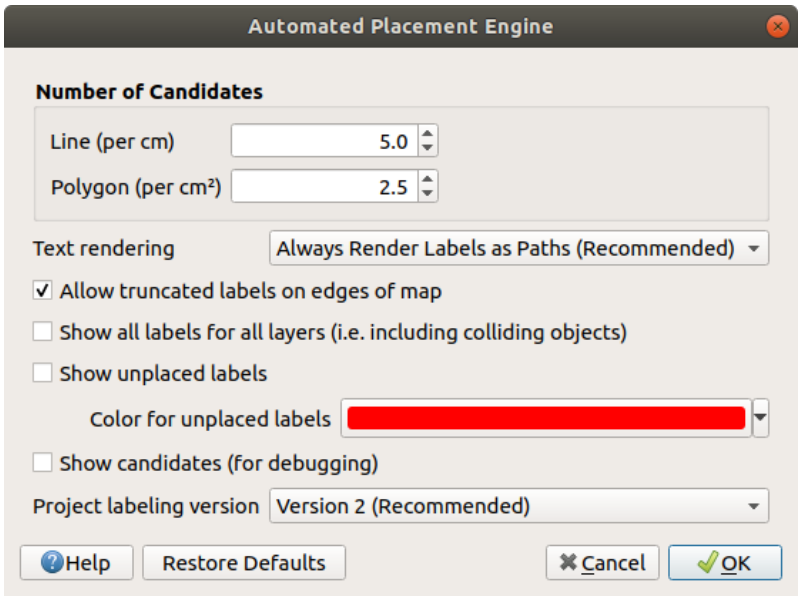


図 16.26: ラベルの自動配置エンジン

- **候補の数** ：ライン地物やポリゴン地物について、地物のサイズに基づき可能なラベル配置の数を計算し割り当てます。地物が長いあるいは広いほど、より多くの候補を持ち、そのラベルは衝突のリスクが減ってより良い配置が可能となります。
- **テキスト出力形式** ：PDF や SVG への **マップキャンバスのエクスポート** や **レイアウトのエクスポート** の際に、ラベルレンダリングウィジェットのデフォルト値を設定します。 **ラベルを常にテキストとしてレンダリング** を選択した場合、ラベルは外部アプリケーション（Inkscape など）で通常のテキストとして編集することが可能です。ただし、副作用としてレンダリング品質が低下し、さらにテキストにバッファ等の特定の設定がなされていると、レンダリングに問題が発生します。このため、ラベルをアウトラインとしてエクスポートする **ラベルを常にパスとしてレンダリング（推奨）** を推奨しています。
- **地図の端のラベルは省略** ：マップ範囲から部分的に外れたラベルをレンダリングするかどうかを制御します。チェックを入れると、（見える範囲内で完全なラベルを配置できない場合、）切れたラベルが表示されます。チェックを外すと、部分的にしか見えないラベルはスキップされます。この設定は、 **地図レイアウトのアイテム** のラベル表示には影響しないことに留意してください。
- **全レイヤの全ラベルを表示（衝突ラベルを含む）** ：このオプションはレイヤ単位で設定することもできます（ **レンダリング** を参照）。
- **位置未定ラベルの表示** ：これにより、重要なラベルがマップから（重複やその他の制約のためなどで）欠落しているかどうかを判断することができます。これらのラベルの表示色はカスタマイズ可能です。
- **候補を表示（デバッグ用）** ：ラベル配置のために生成された候補を示すボックスすべてをマップ上に描画するかどうかを制御します。ラベル名のとおり、これはさまざまなラベル設定がもたらす効果のデバッグとテストのためにのみ有用です。これは、 **ラベルツールバー** のツールを使用して手動でより良い配置を行うのにも便利です。
- **ラベルバージョン** ：QGIS はラベルの自動配置について 2 つの異なるバージョンをサポートしています：
  - *Version 1* ：古いシステム（QGIS3.10 以前のバージョンで使用されており、これらのバージョン

で作成されたプロジェクトを QGIS3.12 以降で開く時に使用）です。 Version 1 では、ラベルや障害物の優先順位を「大まかな目安」としてのみ扱っており、優先度の高い障害物の上に優先度の低いラベルが配置される可能性があります。このため、このバージョンを使用する場合には望むラベル付け結果を得ることが難しい場合があり、古いプロジェクトとの互換性を保つためでしか推奨されません。

- *Version 2* **（推奨）** ：これは、QGIS 3.12 以降で作成された新規プロジェクトでのデフォルトのシステムです。Version 2 では、どんな時にラベルが **障害物** に重なるのを許可するかを決定するロジックが変更修正されました。新しいロジックでは、 ラベル優先度に比べて障害物の重みづけが大きい障害物にラベルが重なることを禁止しています。その結果、このバージョンではラベル付けの結果がより予測しやすく、理解しやすくなりました。

ルールに基づくラベル付け

ルールに基づくラベル付けは **ルールに基づくレンダリング** と同様、複数のラベルの設定を定義し、式フィルタとスケール範囲に基づいて選択的にルールを適用できます。

ルールを作成するには：

1. **ラベル** タブのメインドロップダウンリストで **ルールに基づく定義（rule-based）* オプションを選択します
2. ダイアログ下部にある ルールを追加 ボタンをクリックします
3. 新しく現れたダイアログで、以下を入力します：
   - **説明** ： **ラベル** タブ内でルールの識別に使用するテキストで、印刷レイアウトの凡例の **ラベル凡例エントリ** としても使用します
   - **フィルタ** ：ラベル設定を適用する地物を選択するための式
   - 他のルールがすでに設定されている場合には、 **もしくは** オプションを使用して、同じグループのどのルールのフィルタにもマッチしない地物すべてを選択できます
4. このラベルのルールを適用する **縮尺の範囲** を設定できます。
5. **ラベル** グループボックスで利用できるオプションは、通常の **ラベルの設定** です。ラベルの設定を行い、 *OK* ボタンを押します。

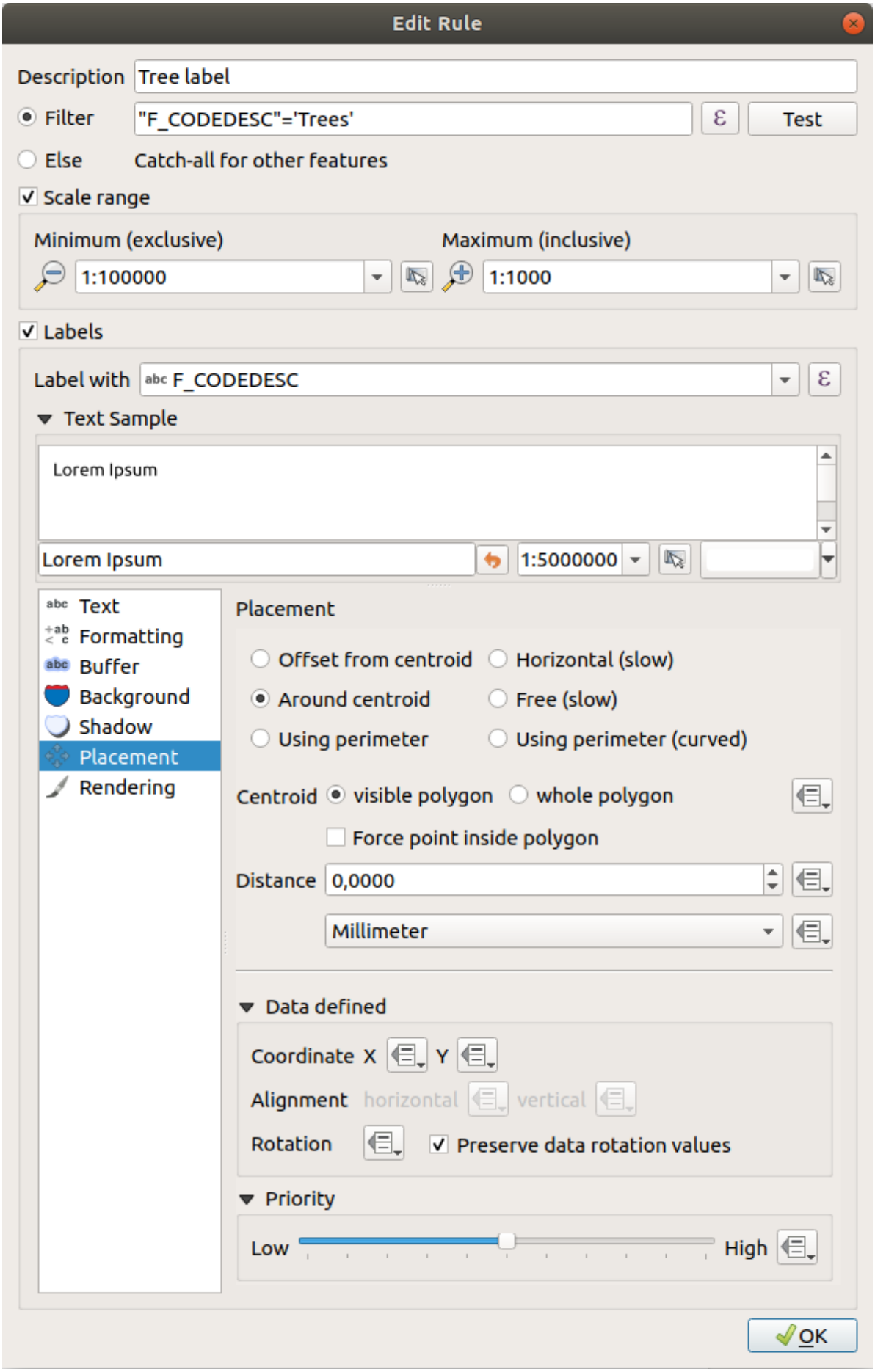


図 16.27: ルールの設定

既存のルールの概要がメインダイアログに表示されます（図 16.28 参照）。複数のルールを追加したり、ドラッグ＆ドロップでルールの並べ替えやルールを入れ子にできます。ルールを ボタンで削除したり、 ボタンまたはダブルクリックでルールを編集できます。

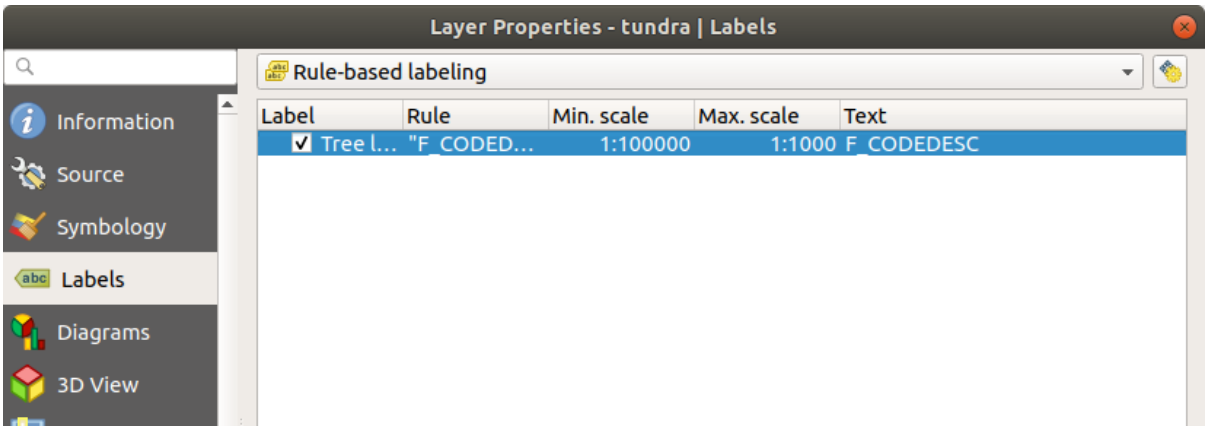


図 16.28: ルールに基づくラベル作成パネル

式に基づいてラベルを定義する

ラベル付けタイプに単一定義、ルールに基づく定義のどちらを選択しても、QGIS では地物のラベル付けに式を使用することができます。

単一定義 を使用している場合には、プロパティのダイアログの **ラベル** タブの **値** ドロップダウンリストの近くにある、 ボタンをクリックします。

図 16.29 に alaska trees レイヤに木のタイプと面積をラベル付けするためのためのサンプル式を示します。'VEGDESC' フィールドと説明用のテキスト、 `$area` に `format_number()` 関数を組み合わせて、ラベルを見栄え良くしています。

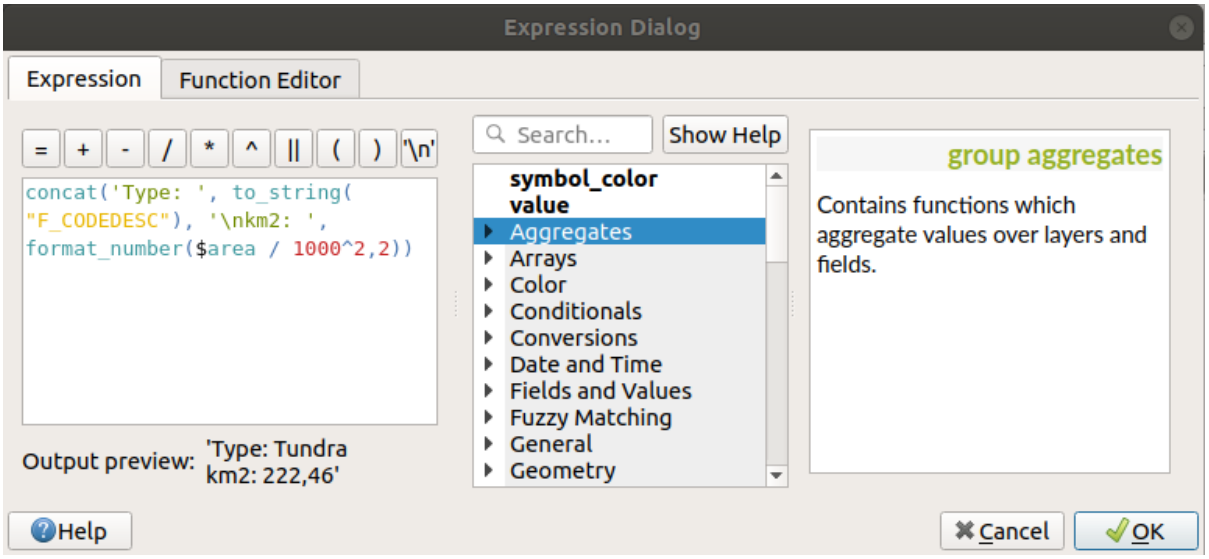


図 16.29: 式を使用したラベル付け

式に基づくラベル付けは簡単です。気を付ける必要があるのは、以下の点だけです。

- すべての要素（文字列、フィールド、関数）を concat や + 、 || のような文字列連結関数で結合する必要があるかもしれません。状況によっては、これらのツールすべてが必要とするものに適合するとは限らないことに注意してください（null や数値が含まれる場合など）。
- 文字列は ' シングルクォート' で書きます。
- フィールドは "ダブルクォート" で書くか、引用符なしで書きます。

いくつかの例を見てみましょう：

1. 2 つのフィールド「name」と「place」をカンマ区切りで並べたラベル:

```
"name" || ', ' || "place"
```

これは次のような文字列を返します:

```
John Smith, Paris
```

2. 2 つのフィールド「name」、「place」と他のテキストによるラベル：

```
'My name is ' + "name" + 'and I live in ' + "place"
'My name is ' || "name" || 'and I live in ' || "place"
concat('My name is ', name, ' and I live in ', "place")
```

これは次のような文字列を返します:

```
My name is John Smith and I live in Paris
```

3. 2 つのフィールド「name」、「place」と他のテキストを様々な連結関数で組み合わせたラベル：

```
concat('My name is ', name, ' and I live in ' || place)
```

これは次のような文字列を返します:

```
My name is John Smith and I live in Paris
```

「place」フィールドが NULL である場合には、次のような文字列が返ります:

```
My name is John Smith
```

4. 2 つのフィールド「name」、「place」と説明テキストによる複数行のラベル：

```
concat('My name is ', "name", '\n' , 'I live in ' , "place")
```

これは次のような文字列を返します:

```
My name is John Smith
I live in Paris
```

5. フィールドと $area 関数を使用した、場所の名前と単位を変換して丸めた面積を表示するラベル：

```
'The area of ' || "place" || ' has a size of '
|| round($area/10000) || ' ha'
```

これは次のような文字列を返します:

```
The area of Paris has a size of 10500 ha
```

6. CASE ELSE 条件を作成します。 *population* フィールドの人口値が <= 50000 であれば town、そうでなければ city とします：

```
concat('This place is a ',
CASE WHEN "population" <= 50000 THEN 'town' ELSE 'city' END)
```

これは次のような文字列を返します:

```
This place is a town
```

7. city の名前を表示し、他の地物はラベルを表示しません（"city" のコンテキストについては、上記の例を参照）：

```
CASE WHEN "population" > 50000 THEN "NAME" END
```

これは次のような文字列を返します:

```
Paris
```

式ビルダーで見ればわかるとおり、データのラベル付けのためにシンプルな式から非常に複雑な式まで作成できる何百もの関数が QGIS にあります。式の詳細と例については **式** の章を参照してください。

ラベル付けのためにデータ定義の上書きを使用する

データによって定義された上書き 機能を使用すると、属性テーブルのエントリやそれに基づく式によってラベル付けの設定が上書きされます。この機能は、上で説明したほとんどのラベル付けオプションの値を設定するために使用することができます。

例えば、Alaska QGIS サンプルデータセットを使用して、 `airports` レイヤを軍用利用の `USE` 列、つまり、空港が以下の意味で利用可能かどうかに基づいて名前をラベル付けしてみましょう：

- 軍が利用可能な場合には、ラベルの色はグレーで、大きさはは 8 です。
- それ以外の場合には、ラベルの色は青で、大きさは 10 です。

このラベル付けを行うためには、レイヤの `NAME` フィールドでのラベル付け（ **ラベルの設定** 参照）を有効にした後に、

1. **テキスト** タブを有効化します。
2. **大きさ** プロパティの横にある アイコンをクリックします。

3. **編集**... を選択し、以下の内容を入力します:

```
CASE
  WHEN "USE" like '%Military%' THEN 8 -- because compatible values are 'Military'
                                      -- and 'Joint Military/Civilian'
  ELSE 10
END
```

4. *OK* ボタンを押して式を有効にします。ダイアログが閉じて ボタンが になり、これはルールが実行されていることを意味します。

5. 次に、色プロパティの横にあるボタンをクリックし、以下の式を入力して有効化します:

```
CASE
  WHEN "USE" like '%Military%' THEN '150, 150, 150'
  ELSE '0, 0, 255'
END
```

同様にして、ラベルの他の任意のプロパティを好きなようにカスタマイズすることができます。 データ定義の上書き ウィジェットの詳細については、 **データによって定義された上書きの設定** セクションの説明と操作方法を参照してください。

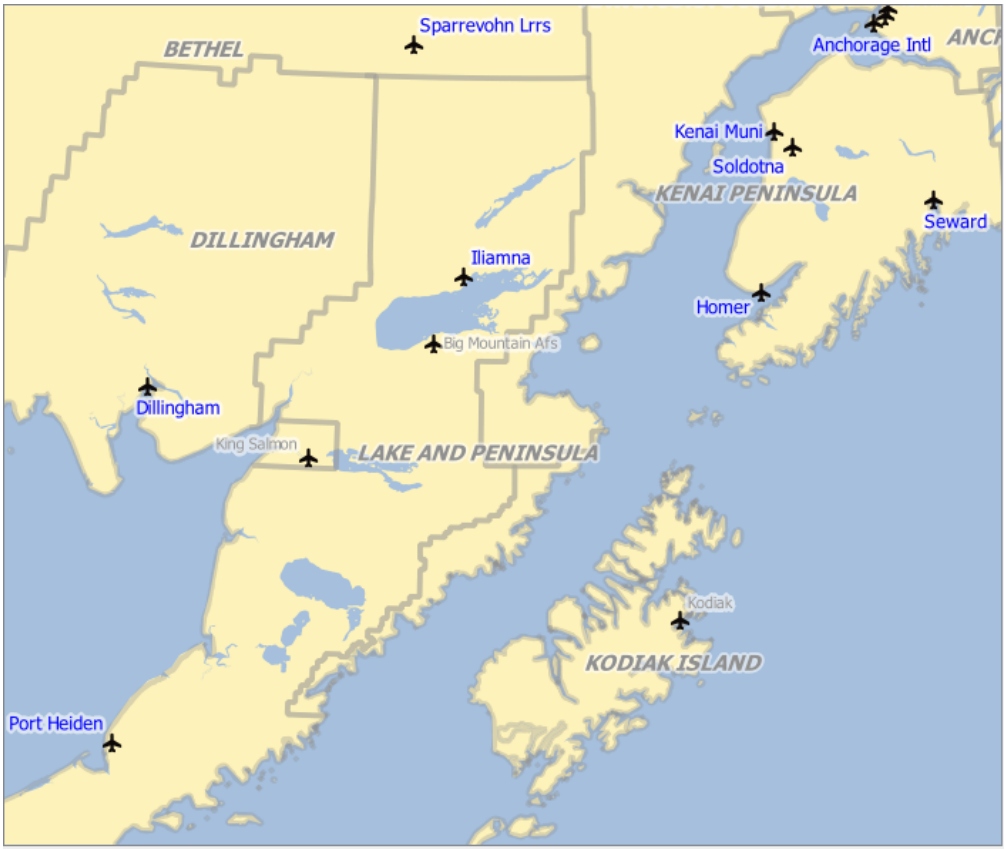


図 16.30: 属性値に基づいて書式設定された空港ラベル

Tip: データ定義の上書きをマルチパート地物の各パートのラベル付けに使用する

ラベルプロパティとは別に、マルチパート地物のラベル付けを設定するオプションがあります。 **レンダリング** を選択し、 地物オプション で **マルチパートの各パートに表示** チェックボックスの横にある データによって定義された上書き ボタンをクリックして、 **データによって定義された上書きの設定** で説明しているようにラベルを定義します。

ラベルツールバー

ラベルツールバー には、 **ラベル** （とその **引出し線** を含む）や **ダイアグラム** のプロパティを操作するためのツールがあります。

図 16.31: ラベルツールバー

- 固定されたラベル・ダイアグラム・引出し線の強調 ：アイテムのベクタレイヤが編集可能である場合には緑色で、そうでない場合は青色で強調表示します。
- 位置未定ラベルの表示切り替え ：重要なラベルが（例えば重なりやその他の制約のために）マップから欠落しているかどうかを確かめることができます。これらのラベルの表示色はカスタマイズ可能です（ **自動配置エンジンの設定** 参照）。
- ラベル・図表の固定/解放 ：アイテムをクリックするか領域をドラッグすると、アイテムを固定します。 Shift キーを押したままアイテムをクリックまたは領域をドラッグすると、ラベルの固定が解放されます。 Ctrl キーを押したままアイテムをクリックまたは領域をドラッグすると、アイテムの固定・解放状態を切り替えることができます。
- ラベル・ダイアグラムの表示/非表示 ：アイテムをクリックするか、 Shift キーを押しながら領域をドラッグすると、アイテムが非表示になります。アイテムが非表示のときは、その地物をクリックするとラベルの表示を元に戻すことができます。領域をドラッグすると、その領域内のすべてのラベル表示が復元されます。
- ラベル、ダイアグラム、引出し線を移動 ：クリックしてアイテムを選択し、好きな位置で再度クリックしてアイテムを移動させます。アイテムの新しい座標が **補助テーブル** に保存されます。このツールでアイテムを選択して Delete キーを押すと、保存された位置の値が消去されます。
- ラベルを回転 ：ラベルをクリックして選択し、好きな回転量で再度クリックすると、回転が適用されます。上と同様で、新しい角度が補助テーブルに保存されます。このツールでラベルを選択して Delete キーを押すと、このラベルの回転の値が消去されます。
- ラベル属性の変更 クリックしたラベルのプロパティを変更するためのダイアログを開きます。このプロパティがあるフィールドにマッピングされていれば、ラベルテキストや座標、回転、フォント、大きさ、複数行の配置などのプロパティを変更することができます。ここでは、 **全パートにラベル** のオプションも設定できます。

警告: ラベルツールによる現在のフィールド値の上書き

ラベルツールバー を使ってラベルをカスタマイズすると、マップされたフィールドにプロパティの新しい値が実際に書き込まれます。したがって、後で必要になるかもしれないデータを誤って置き換えないように注意してください！

注釈: 元となるデータソースを変更せずにラベルをカスタマイズ（位置など）するには、**補助テーブルプロパティ** の機能を使うのがよいでしょう。

マップキャンバスからラベルをカスタマイズする

ラベルツールバー と組み合わせると、データ定義による上書き設定によってマップキャンバス内のラベルを操作（移動、編集、回転）することができます。ここでは、 ラベル、ダイアグラム、引出し線を移動 の機能について、データ定義の上書きを使用した例を説明します（ 図 16.32 参照）。

1. QGIS サンプルデータセットから `lakes.shp` をインポートします。
2. レイヤをダブルクリックしてレイヤプロパティを開きます。 **ラベル** をクリックし、**配置** をクリックしてください。 **重心からのオフセット** を選択します。
3. **データによる定義** エントリを探します。 アイコンをクリックして **座標** ためのフィールドの型を定義します。X には `xlabel` 、Y には `ylabel` を選択してください。するとアイコンが黄色で強調表示されます。

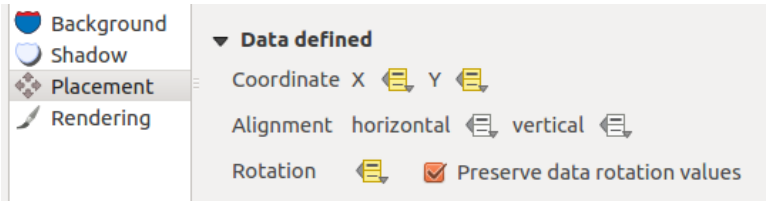


図 16.32: データ定義の上書きを使用したベクタポリゴンレイヤのラベル付け

4. 湖にズームします。
5. 編集モード切替 ボタンを使用して、レイヤを編集可能状態にします。
6. ラベルツールバーの アイコンをクリックします。これでラベルを別の位置に手動で移動させることができます（ 図 16.33 参照）。ラベルの新しい位置は、属性テーブルの `xlabel` と `ylabel` 列に保存されます。
7. 以下の方法で、各湖と移動したラベルとを繋ぐ線を追加することもできます。
   - ラベルの **引出し線プロパティ**
   - 以下の式による **ジオメトリジェネレータシンボルレイヤ**

```
make_line( centroid( $geometry ), make_point( "xlabel", "ylabel" ) )
```

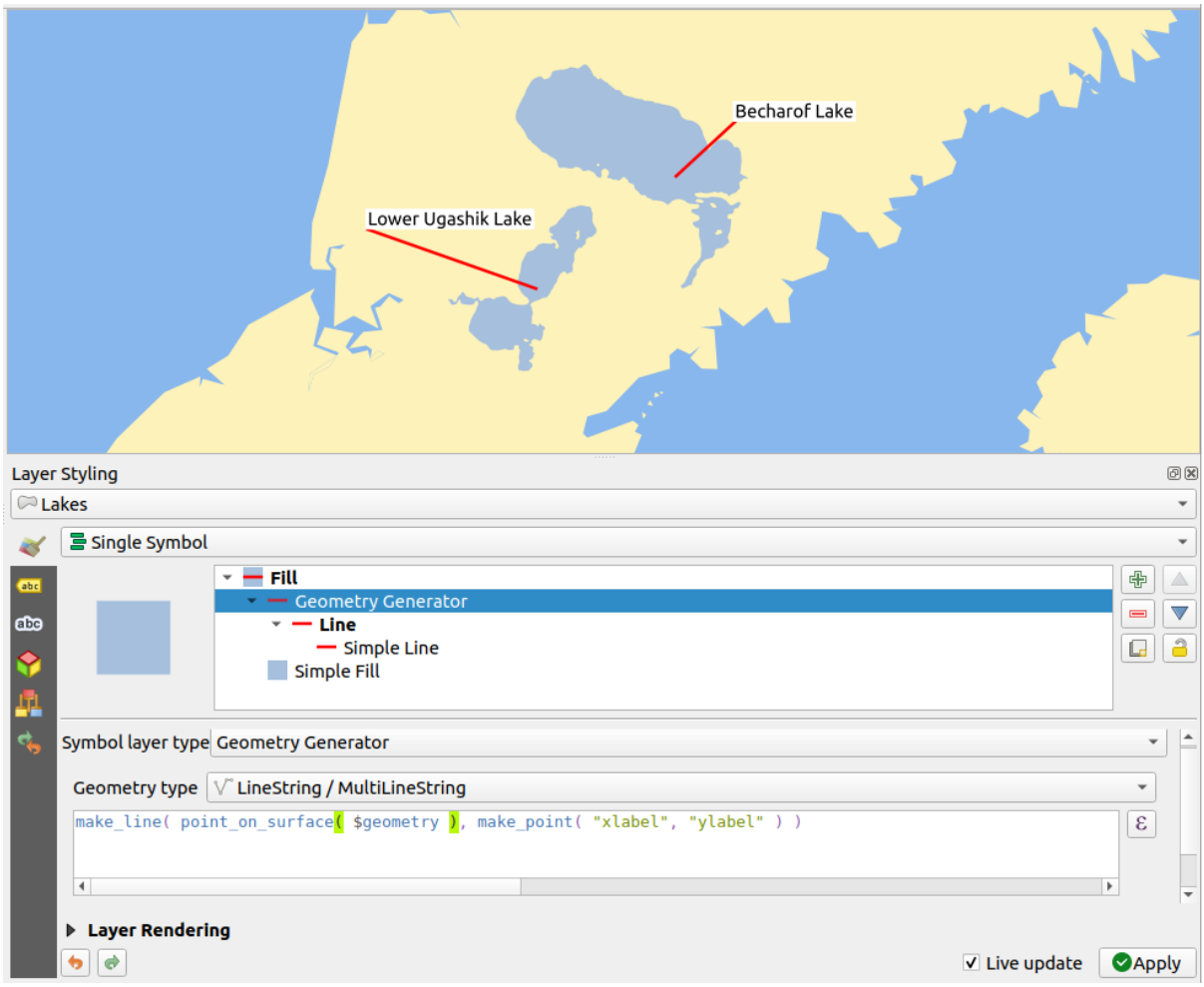


図 16.33: 移動したラベル

注釈: 編集可能なデータソースなしでデータ定義のプロパティを使用するには、**補助テーブルプロパティ**機能を使用するのが良いでしょう。

16.1.5 ダイアグラムプロパティ

ダイアグラム タブでは、ベクタレイヤにグラフィックオーバーレイを追加することができます（図 16.34 参照）。

ダイアグラムのコア実装は現在のところ、以下の機能をサポートしています：

- **ダイアグラムなし**：デフォルト値で、地物上には何のダイアグラムも表示しません。
- **円グラフ**（*Pie Chart*） 円形の統計的な図形で、スライスに分割されており、数値の比率を表します。各スライスの円弧の長さは、それが表す量に比例します。

- abc **テキストダイアグラム** 水平に分割された円で、内部に統計値を表示します。
- **ヒストグラム** 各属性で色の異なるバーがお互いに横に並んだものです。
- **積み上げ棒グラフ** 各属性で色の異なるバーが縦または横に積み重なったものです。

ダイアグラム タブの右上隅には 自動配置設定（すべてのレイヤに適用） ボタンがあり、マップキャンバス上でのダイアグラムの **ラベル配置** を制御するための手段を提供します。

Tip: ダイアグラムの種類をすばやく切り替える

様々なダイアグラムで設定がほぼ共通していることから、ダイアグラムを設計するときには、設定を失うことなくダイアグラムの種類を気軽に変更して、どれがよりデータに適しているかを確認することができます。

ダイアグラムの種類ごとに、プロパティはいくつかのタブに分けられます。

- **属性**
- **レンダリング**
- **大きさ**
- **配置**
- **オプション**
- **凡例**

属性

属性 では、どの変数をダイアグラムに表示するかを定義します。 選択した属性を追加 ボタンを使用して表示したいフィールドを選択し、「利用する属性」パネルに並べます。 **式** を使って生成された属性も使用できます。

任意の行をクリックしドラッグすることで上下に移動させることができ、属性の表示方法を並べ替えられます。また、アイテムをダブルクリックすることで「凡例」列のラベルや属性の色を変更できます。

このラベルは、印刷レイアウトやレイヤツリーの凡例に表示されるデフォルトのテキストです。

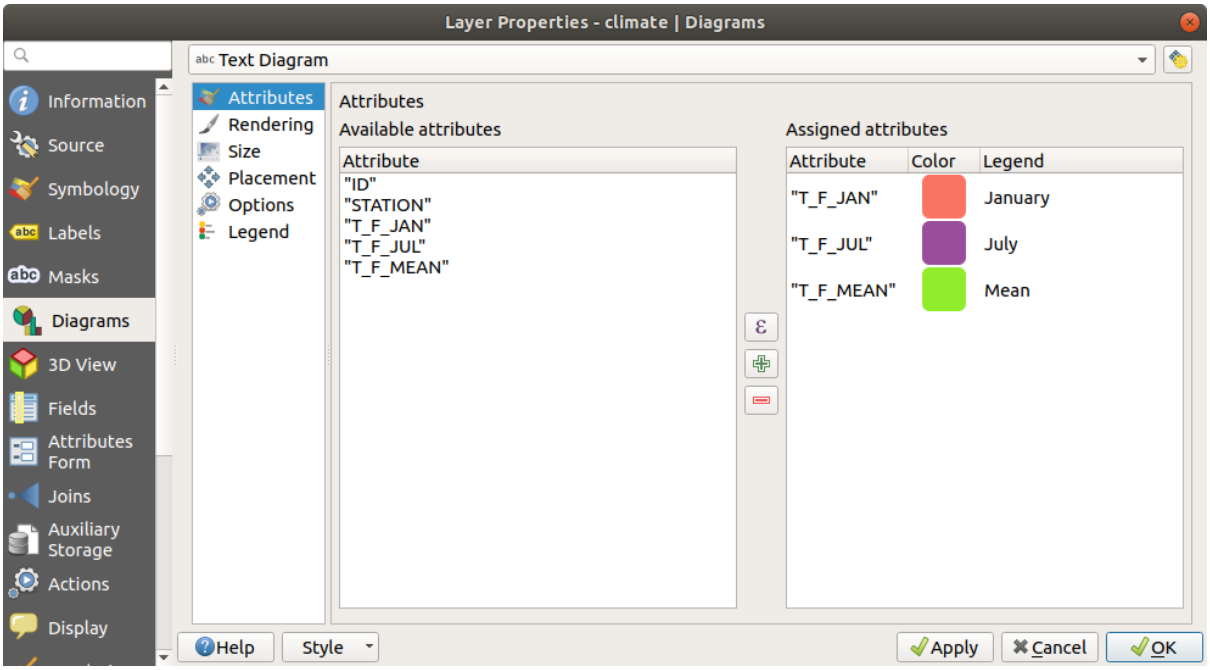


図 16.34: ダイアグラムのプロパティ - 属性タブ

レンダリング

レンダリング は、ダイアグラムの見た目を定義します。ここでは、統計値に干渉されない、以下のような一般的な設定項目があります：

- グラフィックの不透明度、輪郭線の幅、色
- ダイアグラムのタイプによって、以下の設定があります：
  - ヒストグラムと積み上げ棒グラフには、棒グラフの幅とバーの間隔の設定があります。積み上げ棒グラフはバーの間隔を `0` に設定するのが良いでしょう。さらに、**軸の線シンボル** をマップキャンバス上に表示させることができ、**ラインシンボルのプロパティ** を使用してカスタマイズができます。
  - テキストダイアグラムには、円の背景色とテキストに使用される **フォント** の設定があります。
  - 円グラフには、最初の **パイの開始角** と、その **方向** （時計回りか反時計回りか）の設定があります。
- グラフィックに対する **描画エフェクト** の使用

このタブでは、さまざまなオプションを使用してダイアグラムの見た目を管理したり微調整することができます。

- ダイアグラムの *z-index* ：これは、ダイアグラムをお互いに重ねたり、ラベルに重ねたりする方法を制御します。数値の大きいダイアグラムは、他のダイアグラムやラベルの上に描画されます。
- **すべて表示** ：たとえお互いに重なっているとしても、全てのダイアグラムを表示します。
- **表示** ：特定のダイアグラムのみをレンダリングするようにします。

- **常に表示するか否かを示す式** ：たとえ他のダイアグラムや地図ラベルと重なっているとしても常にレンダリングする特定のダイアグラムを選択します。
- **縮尺に応じた表示設定** の設定

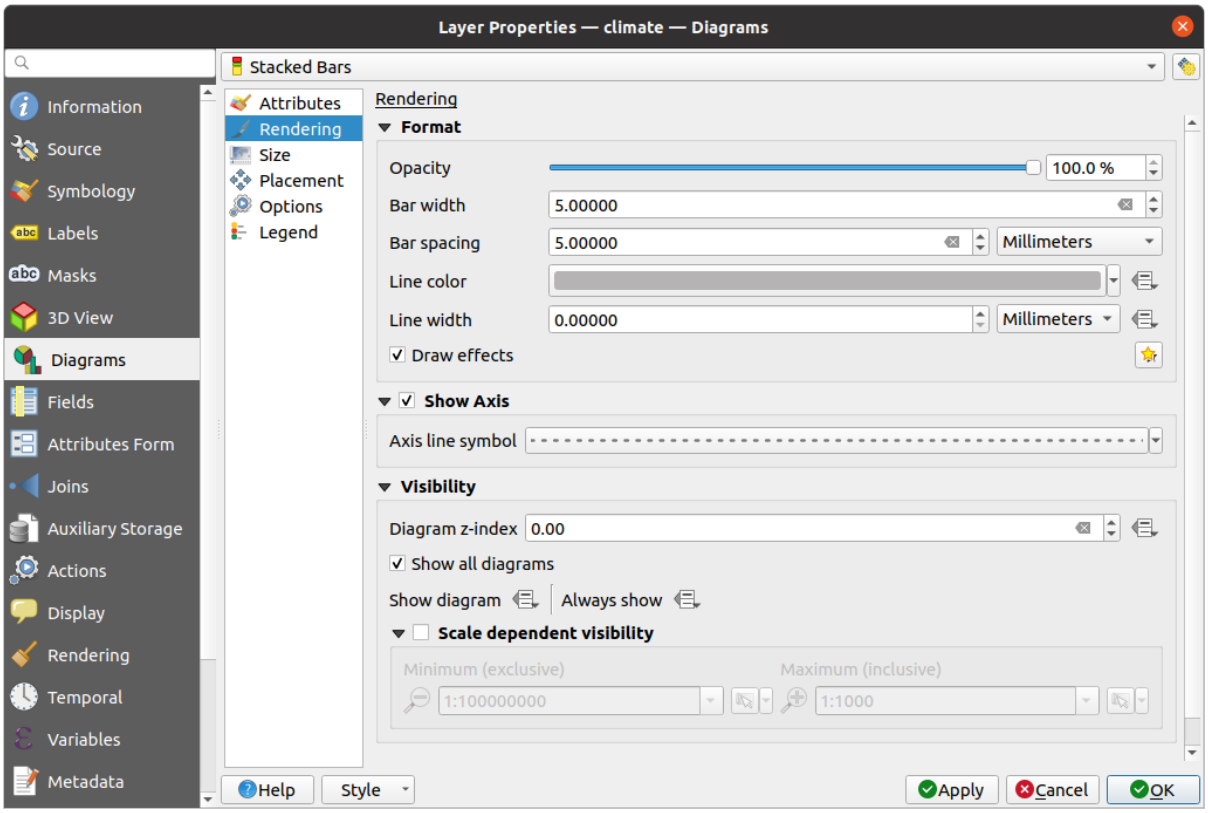


図 16.35: ダイアグラムのプロパティ - レンダリングタブ

大きさ

大きさ は、選択した統計情報の表示方法を設定するメインタブです。ダイアグラムのサイズの **単位** は、「ミリメートル」、「ポイント」、「ピクセル」、「地図単位」、「インチ」のいずれかです。サイズの設定には以下のものを使用できます：

- **固定サイズ** ：すべての地物のグラフィックを表現するためにただ一つだけのサイズを使用します（ヒストグラムでは使用できません）。
- **可変サイズ** ：レイヤの属性を使用した式に基づいたサイズを使用します:

  1. **属性** で、フィールドを選択するか式を構築します
  2. **検索** ボタンを押して属性値の **最大値** を求めるか、このウィジェットにカスタム値を入力します。
  3. ヒストグラムと積み上げ棒グラフでは、属性の **最大値** に対応する **バーの長さ** の値を入力します。各地物について、バーの長さはこの対応関係を維持するために線形にスケーリングされます。
  4. 円グラフとテキストダイアグラムでは、属性の **最大値** に対応する **大きさ** の値を入力します。各地物について、円の面積や直径はこの対応関係を維持するために（ 0 から）線形にスケーリングされます。ただし、小さなダイアグラムには **最小サイズ** を設定することができます。

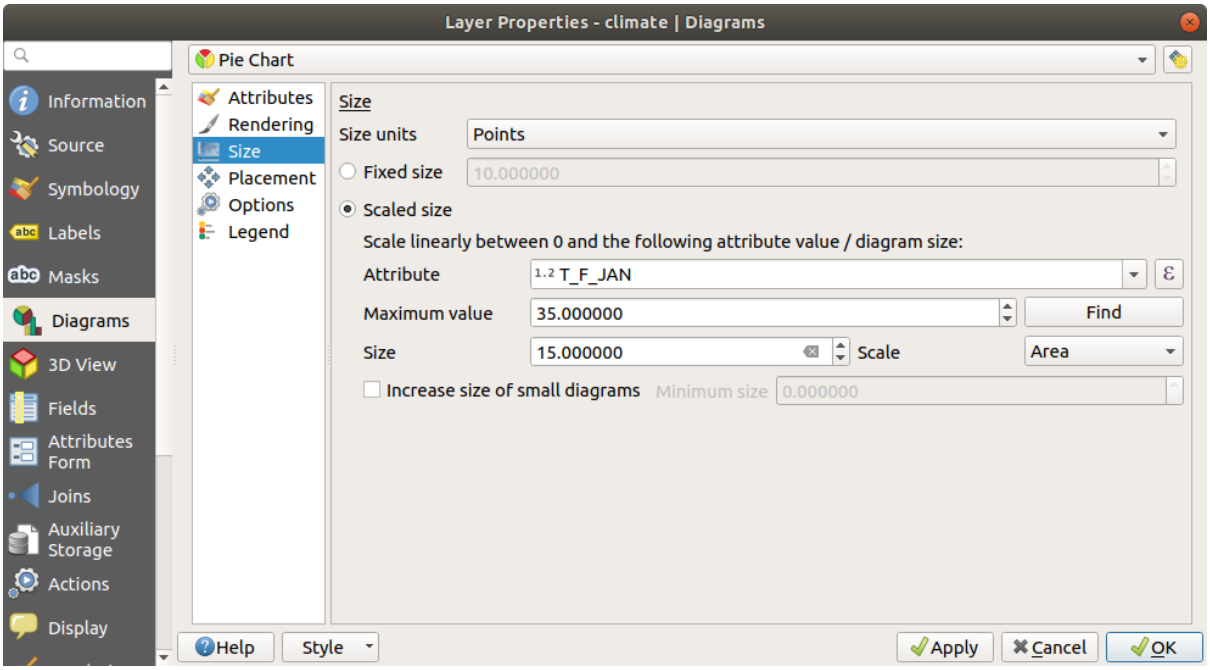


図 16.36: ダイアグラムのプロパティ - 大きさタブ

配置

配置 は、ダイアグラムの位置を定義します。レイヤのジオメトリタイプに応じて、配置のための様々なオプションがあります（詳細は **配置** を参照）：

- ポイントジオメトリに対しては、 **点の周り** または **点に重ねる** オプションがあります。前者には、守るべき距離の設定が必要です。
- ラインジオメトリに対しては、 **線の周り** または **線に重ねる** オプションがあります。ポイント地物同様、一つ目のオプションには守るべき距離の設定が必要であり、地物に対するダイアグラムの配置を指定することができます（「線の上」、「線上」または「線の下」）。同時に複数のオプションを選択することが可能です。その場合、QGIS はダイアグラムの最適な位置を探します。ダイアグラムの位置のために線の方向を利用することもできることを覚えておいてください。
- ポリゴン地物には、（設定した **距離** 離れた） **重心の周り** 、 **重心に重ねる** 、 **周辺を利用** 、 **ポリゴンの内側** のオプションがあります。

座標 グループでは、属性値や式を使用して *X* 、 *Y* 座標を設定することで、地物ごとの基準でダイアグラムの配置を直接制御することができます。この情報は **ラベル、ダイアグラムを移動** ツールを使用しても入力されます。

優先度 セクションでは、各ダイアグラムの配置優先度を定義することができます。すなわち、同じ場所に異なるダイアグラムやラベルの候補がある場合、優先度の高いアイテムが表示され、他のアイテムは省略されます。

ダイアログやラベルを地物と重ならないようにするか否かを示す式 は、地物を **障害物** として使用するかを定義します。すなわち、 QGIS は障害物として使用する地物上にラベルやダイアグラムが重ならないように配置しようとします。優先度はこの時に、より重みの大きな地物のためにダイアグラムが非表示となるかどうかを評価するために使用されます。

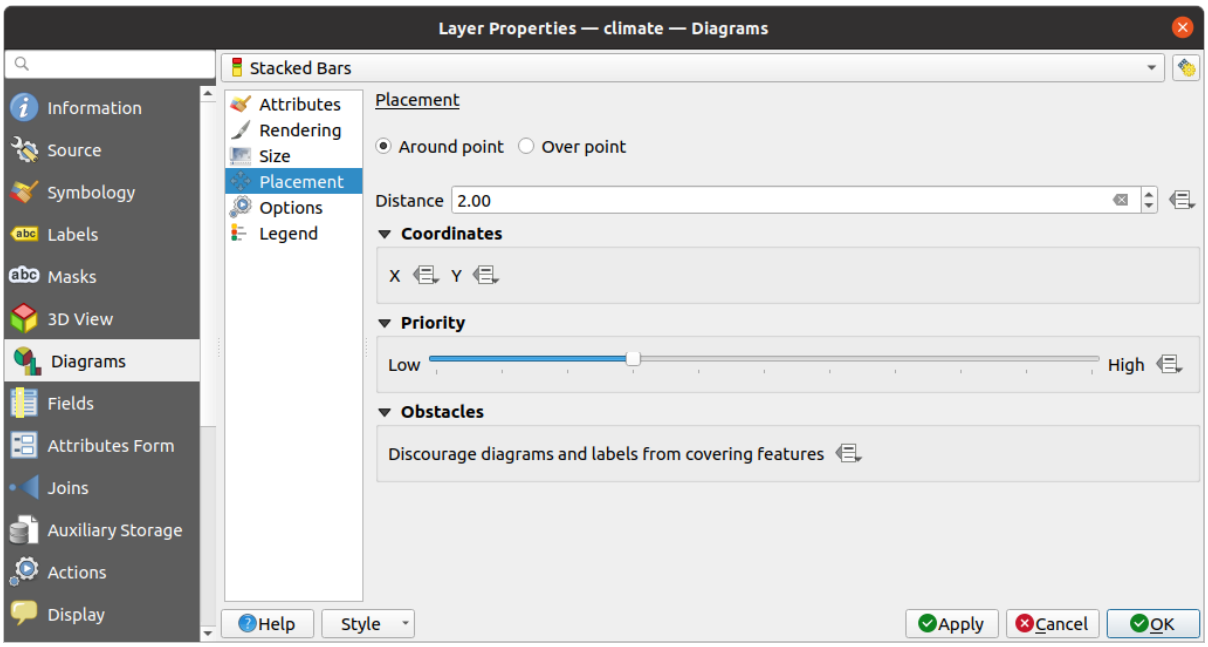


図 16.37: ベクタプロパティダイアログのダイアグラムプロパティの配置タブ

オプション

ヒストグラムと積み上げ棒グラフには **オプション** タブに設定があります。水平あるいは垂直方向のダイアグラムについて、 **棒グラフの向き** を **上** 、 **下** 、 **右** 、 **左** から選択することができます。

凡例

凡例 タブでは、 **レイヤパネル** や **印刷レイアウトの凡例** 内に、レイヤのシンボロジに続けてダイアグラムのアイテムを表示するかを選択することができます。

- **ダイアグラムの凡例を表示** にチェックを入れると、上の **属性** タブで設定した 色 と 凡例 プロパティを凡例内に表示します。
- そして、ダイアグラムに **可変サイズ** を使用している場合には、 **可変サイズダイアグラムの凡例...** ボタンを押すと、凡例内でのダイアグラムのシンボルの比率を設定できます。ボタンを押すと **データによるサイズ定義** ダイアログが開き、 **データ定義による凡例サイズ** で説明したオプションがあります。

設定が済んだら、レイヤシンボロジに続けてダイアグラムの凡例アイテム（属性名と色、ダイアグラムサイズ）も印刷レイアウトの凡例に表示されるようになります。

16.1.6 マスクプロパティ

マスク タブでは、任意の他のシンボルレイヤやラベルと重なっている現在のレイヤのシンボルの設定ができます。これは、色が似ていて重なった場合に判読の難しいシンボルとラベルの読みやすさを向上させるためのものです。現在のレイヤのシンボルレイヤの一部を「隠す」ために、アイテムの周りにカスタムの透明なマスクを追加します。

アクティブレイヤにマスクを適用するには、まず初めにプロジェクトで **マスクシンボルレイヤ** または **マスクラベル** を有効にする必要があります。それから、 **マスク** タブにおいて以下の項目にチェックを入れます：

- **マスクするシンボルレイヤ** ：現在のレイヤの全てのシンボルレイヤがツリー構造でリストされています。ここでは、選択したマスクソースと重なった場合に透過的に「切り抜き」たいシンボルレイヤのアイテムを選択することができます。
- **マスクソース** ：プロジェクト内で定義されている全てのマスクラベルとマスクシンボルがリストされています。選択したマスクするシンボルレイヤに対してマスクを生成したいアイテムを選択します。

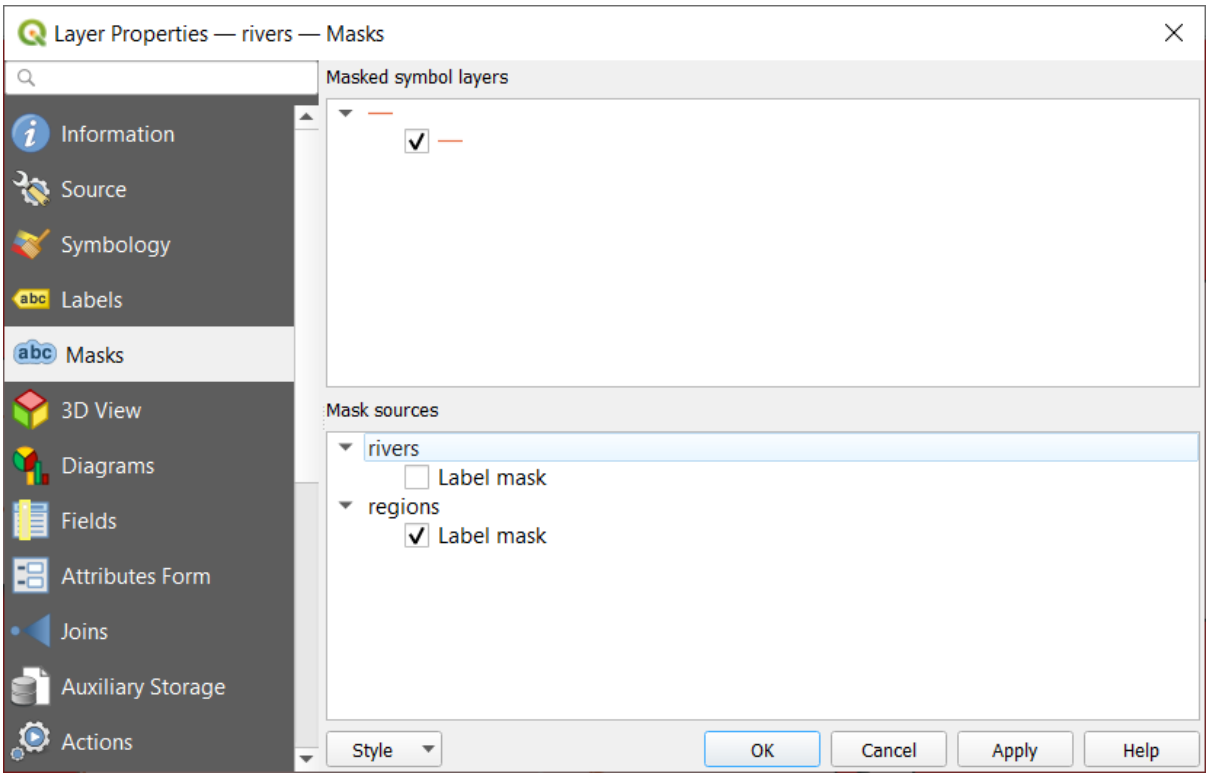


図 16.38: レイヤプロパティ - マスクタブ

16.1.7 3D ビュープロパティ

3D **ビュー** タブでは、 *3D* **マップビュー** ツールで描画されるベクタレイヤに対する設定ができます。

レイヤを 3D で表示するには、タブの上部にあるコンボボックスから以下のどちらかを選択します：

- **単一定義** ：地物は共通の 3D シンボルを使用してレンダリングされます。プロパティは **データ定義** とすることもできます。レイヤの各ジオメトリタイプに対する詳細については、 *3D* **シンボルの設定** を参照してください。
- **ルールによる定義** ：複数のシンボル設定を定義し、式フィルタや縮尺範囲に応じて選択的に適用することができます。使い方の詳細については、 **ルールベースのレンダリング** を参照してください。

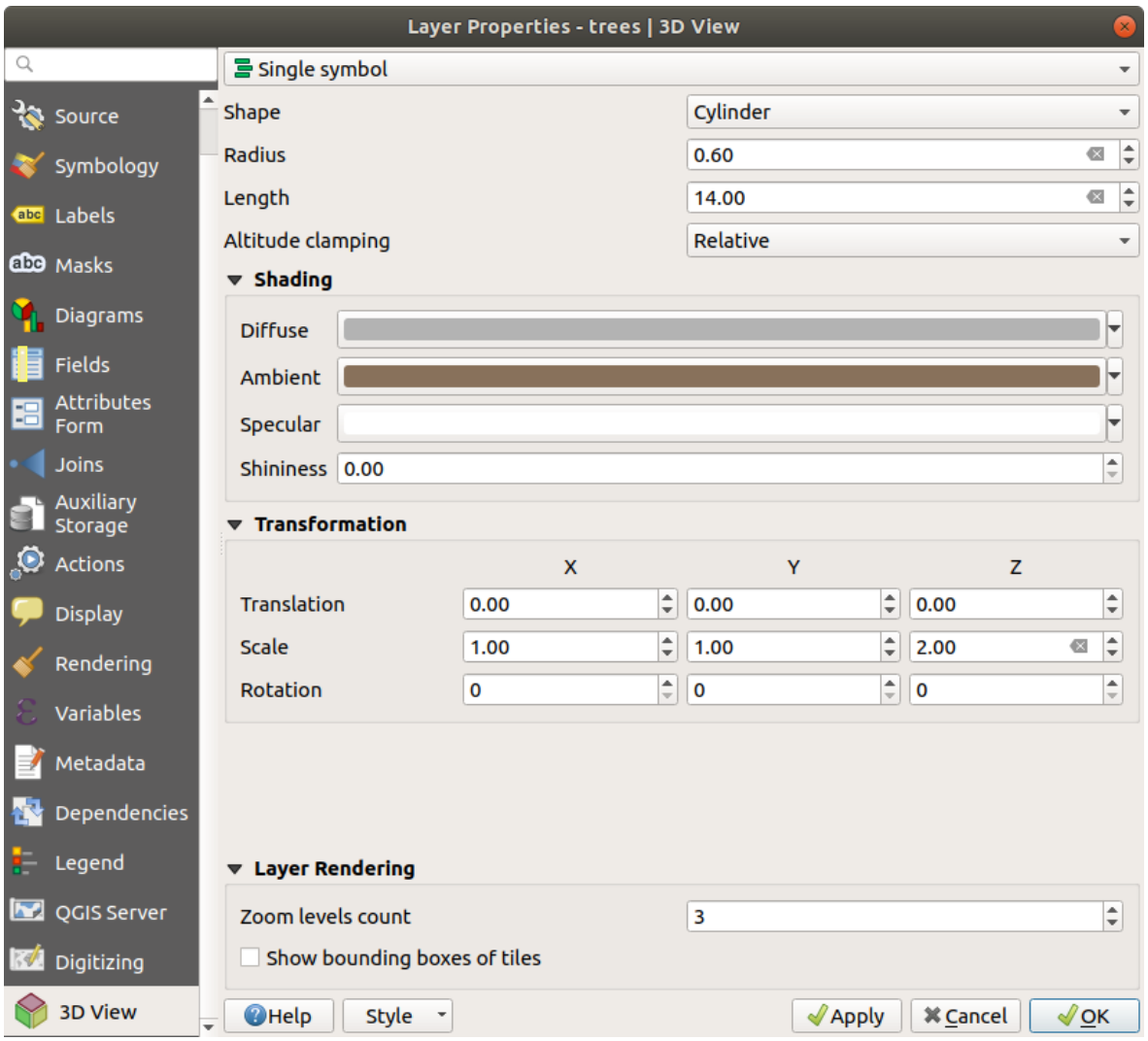


図 16.39: ポイントレイヤの 3D プロパティ

注意: シンボルの標高と地形の設定 は、**標高** タブが **望ましい**

3D **ビュー** タブにある地物の標高と高度に関連するプロパティ（**標高の制約**、 **標高の拘束**、 **押し出し** または **高さ**）は、レイヤの **標高プロパティ** からデフォルト値を継承するので、 **標高** タブ内で設定す

ることが望ましい。

パフォーマンス向上のため、ベクタレイヤからのデータはマルチスレッドを使ってバックグラウンドで読み込まれ、タイルでレンダリングされます。タイルのサイズはタブの **レイヤレンダリング** セクションで制御できます：

- **ズームレベルカウント** ：四分木の深さを決定します。例えば、ズームレベル 1 は、レイヤ全体に対してただ一つのタイルがあることを意味します。ズームレベル 3 は、リーフレベルに 16 枚のタイルがあることを意味します（ズームレベルが増えるごとに 4 倍になります）。デフォルト値は 3 で、最大値は 8 です。
- **タイルのバウンディングボックスを表示** ：表示されるべきタイルが表示されない問題がある場合に特に有用なオプションです。

16.1.8 属性プロパティ

属性 タブでは、レイヤに関連したフィールドの情報が表示され、それらを編集することもできます。レイヤは 編集モード切替 ボタンを使用して **編集可能** にすることができます。編集モードでは、新規フィールド と フィールド削除 ボタンを使用して、フィールド構造を変更することができます。

また、フィールド名をダブルクリックすることで名前の変更もできます。これは PostgreSQL、Oracle やメモリレイヤおよび GDAL のバージョンに依存した、一部の GDAL レイヤでのみサポートされています。

データソースや **属性フォームプロパティ** の設定では、フィールドの別名も表示されます。別名とは、地物のフォームや属性テーブルに使用することができる、人間が読みやすいフィールド名のことです。別名はプロジェクト内に保存されます。

データセットに含まれているフィールドや **仮想フィールド** 、 **補助テーブル** の他に、 **フィールド** タブには **結合したレイヤ** のフィールドもリストされます。フィールドの出典に応じて、フィールドには異なる背景色が適用されます。

リストされた各フィールドについて、 **データ型** 、 **タイプ名** 、 **長さ** や **精度** `といった読取専用の属性もダイアログに表示されます。

データプロバイダによっては、例えばフィールド作成時にコメントをフィールドに関連付けることができます。この情報は読み込まれて **コメント** 列に表示され、地物フォームでフィールドのラベルにカーソルを乗せた際に表示されます。

設定 列では、特定の状況でのフィールドの振る舞いを設定できます：

- `Not searchable` : **検索ロケータバー** による検索を行いたくない場合には、このオプションにチェックを入れます
- `WMS` 経由で公開しない ：QGIS サーバーで WMS としてこのレイヤを提供する場合に、このフィールドを表示したくない場合には、このオプションにチェックを入れます
- `WFS` 経由で公開しない ：QGIS サーバーで WFS としてこのレイヤを提供する場合に、このフィールドを表示したくない場合には、このオプションにチェックを入れます

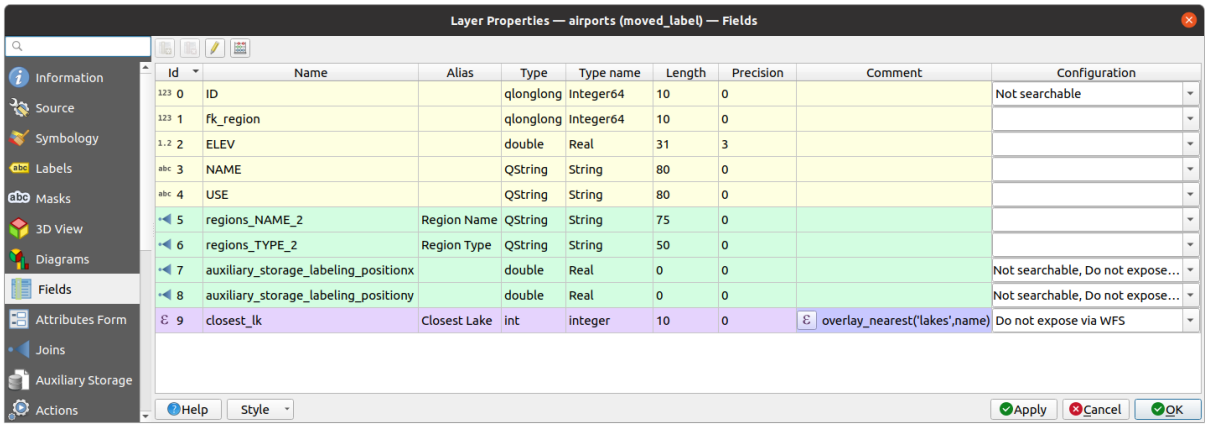


図 16.40: 属性プロパティタブ

16.1.9 属性フォームプロパティ

属性フォーム タブでは、新しい地物を作成したときや既存の地物に対するクエリを行ったときに表示されるフォームの設定ができます。以下のものが定義できます：

- 地物フォームや属性テーブルの各フィールドの見た目や動作（ラベル、ウィジェット、制約条件など）
- フォームの構成（カスタムもしくは自動生成）
- フォームやフィールドウィジェットとのインタラクションを処理するための Python による追加ロジック

ダイアログの右上では、新しい地物を作成する際にフォームがデフォルトで開くかどうかの設定ができます。この設定はレイヤ毎に設定することもできますし、**設定** **オプション** **デジタイズ** メニューの **地物作成後に属性フォームをポップアップさせない** オプションでグローバルに設定することもできます。

データに合わせてフォームをカスタマイズする

デフォルトでは、 地物情報表示 ツールで地物をクリックするか属性テーブルを **フォーム表示** モードに切り替えると、QGIS は定義済みのウィジェット（通常はスピンボックスやテキストボックスで、各フィールドは専用の行として表現され、ウィジェットの横にはラベルがついています）を持った基本的なフォームを表示します。レイヤに **テーブル結合** が設定されている場合には、参照しているレイヤのフィールドは同じ基本構造に従って、フォームの下部にある埋め込みフレームに表示されます。

この表現は **レイヤプロパティ** **属性フォーム** タブの **属性レイアウトエディタ** の設定値をデフォルトの 自動生成 とした場合の結果です。このプロパティは 3 つの値をとります：

- 自動生成 ：フォームは「1 行に 1 フィールド」の基本構造を取りますが、対応する各ウィジェットはカスタマイズも可能です。
- ドラッグ＆ドロップデザイナー ：ウィジェットのカスタマイズのほかに、例えばウィジェットをグループやタブに埋め込むことによって、フォームの構造をより複雑にできます。
- ui ファイルを提供する ：地物フォームとして Qt designer ファイルを使用することで、より複雑で完全な機能を持ったテンプレートになります。

自動生成フォーム

自動生成 オプションがオンになっている時は、 **利用可能なウィジェット** パネルには、フォームに表示されるフィールド（レイヤのものと結合レイヤのもの）がリスト表示されています。フィールドを選択して、右側パネルでウィジェットの見た目や振る舞いを設定できます：

- フィールドに **ラベルのカスタマイズや自動的なチェック** を追加
- **特定のウィジェット** を使用するように設定

ドラッグ＆ドロップデザイナー

ドラッグ＆ドロップデザイナーは、例として 図 16.41 に示すように、属性フィールドを表現するためのさまざまなコンテナ（タブやグループ）や、特定のフィールドと直接にはリンクしていないその他のウィジェット（HTML/QML ウィジェットやレイヤに定義された **アクション** 等）を持ったフォームを作成できます。

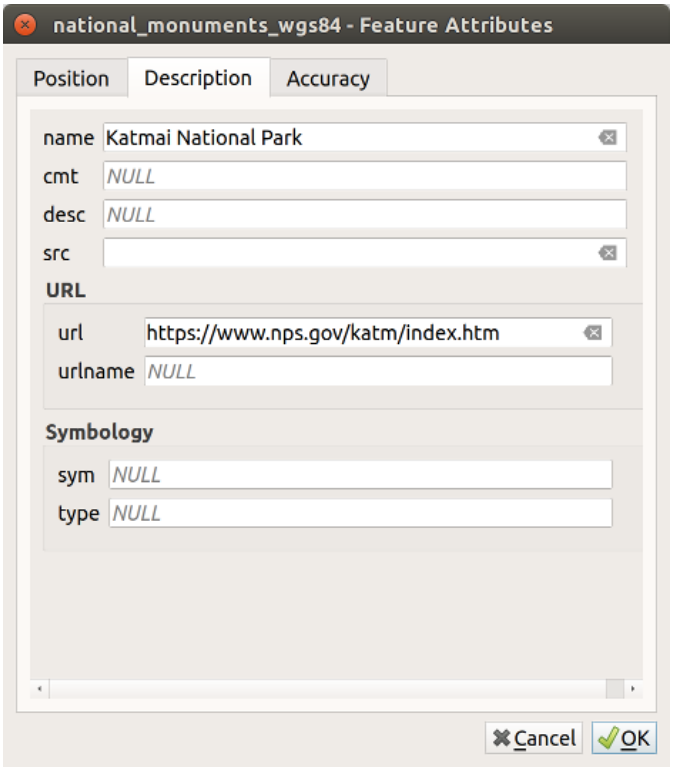


図 16.41: タブと名前付きグループを備えた組み込みフォームの作成結果

1. **属性レイアウトエディタの選択** コンボボックスから ドラッグ＆ドロップデザイナー を選択します。これにより **利用可能なウィジェット** パネルの隣に **フォームのレイアウト** パネルが利用できるようになります。パネルには既存のフィールドが表示されています。フィールドを選択すると、3 つ目のパネルに **プロパティ** が表示されます。

2. **フォームのレイアウト** パネルで使用したくないフィールドは、選択して ボタンを押すことで削除できます。 選択部分を反転する ボタンを押すと、選択・未選択状態を切り替えることもできます。

3. 最初のパネルから **フォームのレイアウト** パネルにフィールドをドラッグ＆ドロップすることで、フィールドを再追加できます。同じフィールドを複数回追加することができます。

4. **フォームのレイアウト** パネル内でフィールドをドラッグ＆ドロップすることで位置の並べ替えができます。

5. コンテナを追加して同じカテゴリに属するフィールドを関連付け、フォームの構造を改良します。

   1. 最初に フォームレイアウトに新しいタブまたはグループを追加します アイコンを使います。フィールドや他のグループがその中に表示されます。

   2. 次に、以下のコンテナのプロパティを設定します：

      - **ラベル**: コンテナに使用されるタイトル
      - **コンテナ型**: これは **タブ** または **コンテナのグループボックス**（タブまたは別のグループ内の折りたたみ可能なグループボックス）になります
      - 埋め込みフィールドを並べる **列の数**

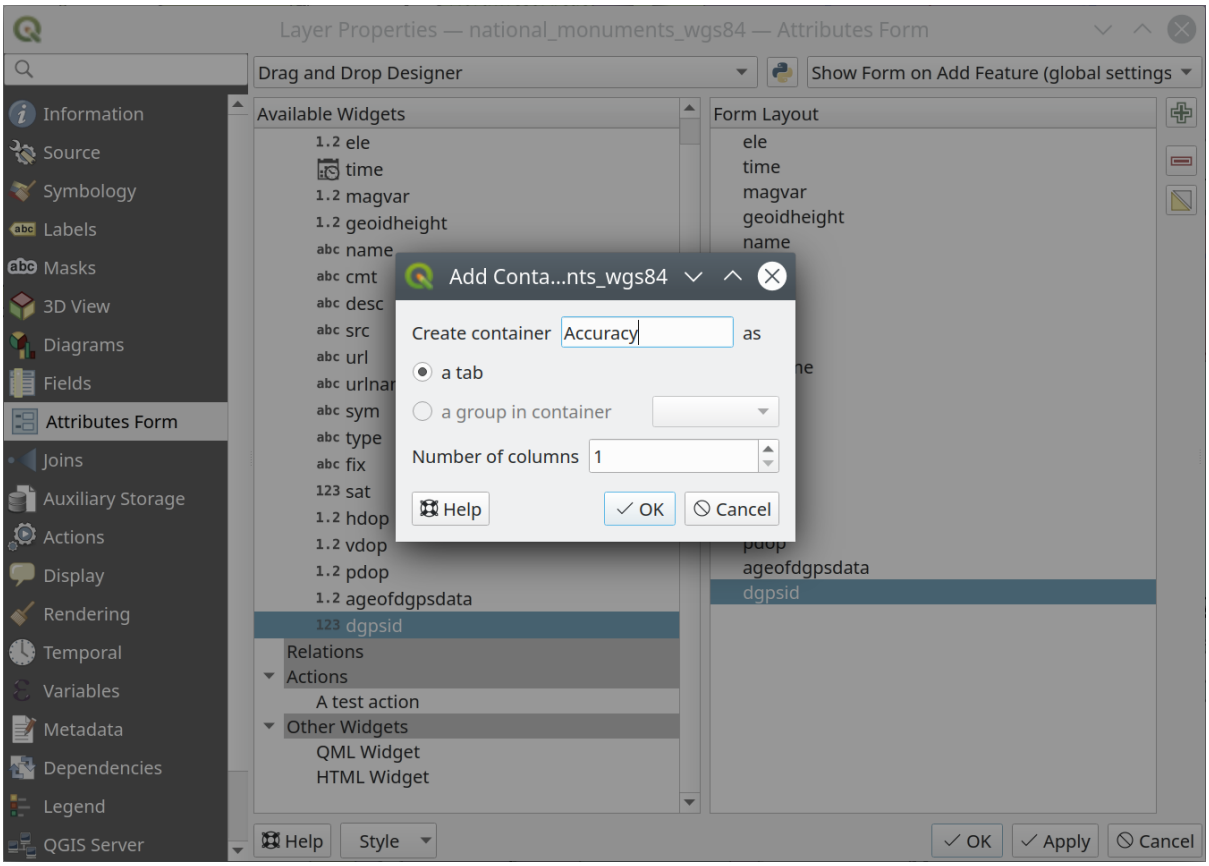


図 16.42: **属性レイアウトエディタ** のコンテナ作成ダイアログ

以下に挙げるものやその他のプロパティは、アイテムを選択して 3 つ目のパネルで後からでも更新が可能です：

- コンテナのラベルの表示・非表示
- コンテナの名前の変更
- カラムの数の設定

- コンテナの表示を制御する式の入力。この式はフォームが変更されるたびに毎回再評価され、これに応じてタブやグループボックスが表示 / 非表示になります。
- **グループボックスとして表示**: タブをトップレベルのグループボックスに変換します
- グループボックスの場合、フォームを開いたときに、すべての地物に対して *Collapsed* して表示するか、式にマッチする地物に対してのみ表示（**式によって折り畳みを制御**）するかを設定します。
- コンテナの **スタイル** の外観を、カスタム **背景色**、ラベルの色、フォントのプロパティで設定します

3. フォームレイアウトに新しいタブまたはグループを追加します アイコンをもう一度押すと、好きなだけコンテナを作成し、埋め込むことができます。

6. 次のステップとして、単純なドラッグ・アンド・ドロップで、関連するフィールドを各コンテナに割り当てます。グループボックスやタブも同じ方法で移動できます。
7. 使用するフィールドの **ウィジェットをカスタマイズ する**
8. レイヤが **１対多または多対多関係** にある場合、**利用可能なウィジェット** パネルから **フォームのレイアウト** パネルにリレーション名をドラッグ＆ドロップしてください。関連するレイヤの属性フォームが現在のレイヤのフォームの選択された場所に埋め込まれます。他のアイテムと同様、プロパティの設定を行うには、リレーションのラベルを選択します：
   - リレーションのラベルの表示・非表示
   - リンクボタンの表示
   - リンク解除ボタンの表示
9. レイヤが レイヤ や 地物 のスコープで有効な一つ以上の **アクション** を持つ場合には、**アクション** の下にアクションが表示され、その他のフィールドと同様にドラッグ＆ドロップできます。関連付けられたアクションは、現在のレイヤのフォームの指定した場所に埋め込まれます。
10. さらに **他のウィジェット** から一つ以上のウィジェットを追加してフォームをカスタマイズします（**他のウィジェット** を参照）
11. レイヤのプロパティダイアログの適用ボタンを押します。
12. 地物属性フォームを開く（例えば 地物情報表示 ツール）と、新しいフォームが表示されます。

他のウィジェット

ドラッグ＆ドロップデザイナーには、特定のフィールドに接続されていないウィジェットが多数用意されています。これらはフォームの外観を向上させたり、動的に計算された値を表示するために使用できます。

- *HTML* **ウィジェット**: HTML ページを埋め込みます。その HTML ソースは、動的に計算された式の結果を含めることができます。
- *QML* **ウィジェット**: QML ページを埋め込みます。その QML ソースは動的に計算された式の結果を含めることができます。

カスタム ui ファイルを使用する

ui ファイルを提供する オプションを使うと、Qt デザイナーで作られた複雑なダイアログを使用できます。ui ファイルを使用することで、ダイアログボックスの作成がかなり自由に行えます。なお、グラフィカルオブジェクト（テキストボックス、コンボボックス等）をレイヤのフィールドにリンクさせるためには、同じ名前を与える必要があるということに注意してください。

UI **の編集** を使って、使用するファイルへのパスを定義します。

ui ファイルはリモートサーバーでホストすることも可能です。この場合には、 *UI* **の編集** にはファイルパスではなく、フォームの URL を指定します。

QGIS-training-manual-index-reference 内の 新しいフォームの作成 レッスンにいくつかの例があります。更なる詳細情報については、 https://woostuff.wordpress.com/2011/09/05/qgis-tips-custom-feature-forms-with-python-logic/ を参照してください。

カスタム関数でフォームの機能を強化する

QGIS のフォームには、ダイアログを開いたときに呼び出される Python の関数を持たせることができます。ダイアログに追加のロジックを加えるには、この関数を使用します。フォームのコードは 3 つの異なる方法で指定できます：

- 環境変数から読み込む ：例えば `startup.py` 内の関数やインストールされたプラグインの関数を使用します
- 外部ファイルから読み込む ：ファイルブラウザを使用してファイルシステムから Python ファイルを選択するか、リモートファイルの URL を入力します。
- このダイアログでコードを入力 ：Python エディタが表示され、使用する関数を直接入力できます。

すべての場合において、呼び出される関数の名前を入力する必要があります（下の例では *open* ）。

例（MyForms.py モジュール）：

```
def open(dialog,layer,feature):
    geom = feature.geometry()
    control = dialog.findChild(QWidget,"My line edit")
```

Python の init 関数での参照は open のようになります。

フィールドの動作を設定する

属性フォーム タブのメイン部分では、地物テーブルや地物フォームにおいてフィールドの値の入力や表示に使用されるウィジェットのタイプを設定することができます。ユーザーが各フィールドをどのように操作するかを定義したり、各フィールドに入力できる値や範囲を定義することができます。

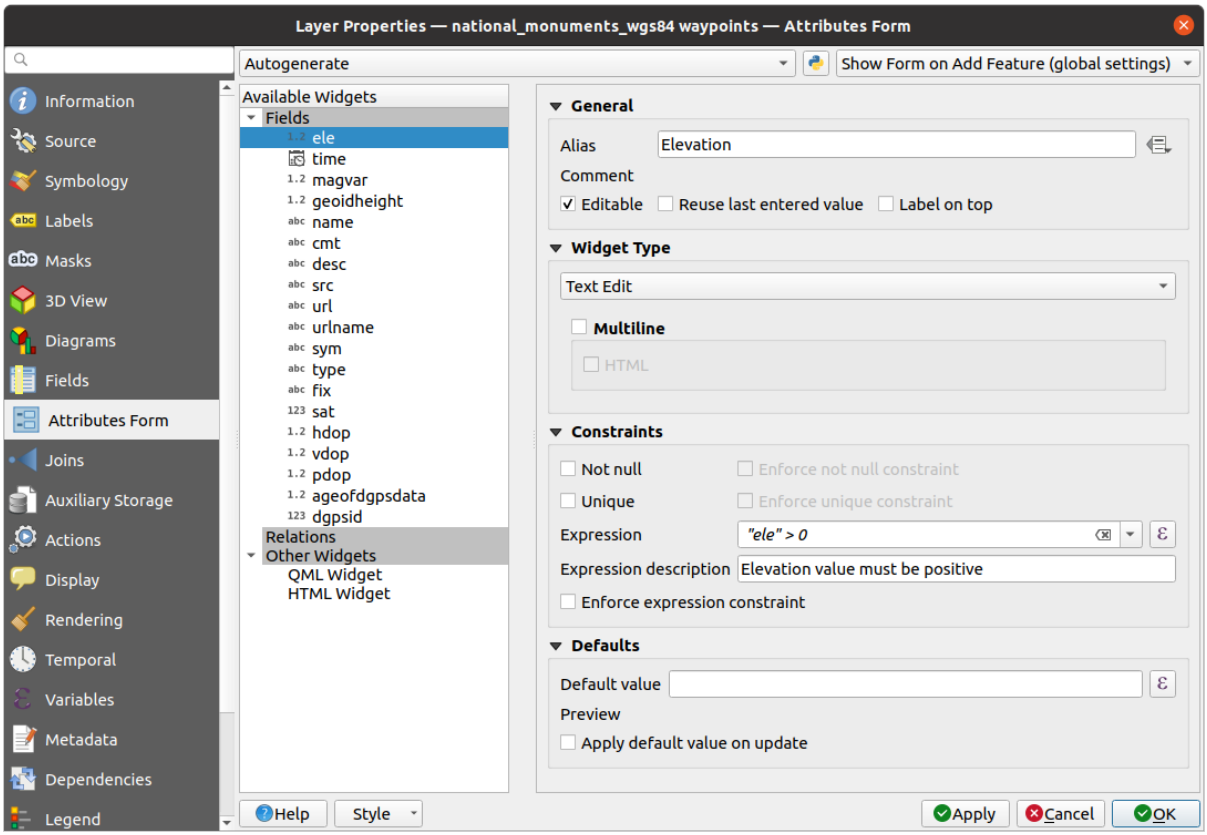


図 16.43: 属性カラムの編集ウィジェットを選択するためのダイアログ

共通設定

フィールドに適用するウィジェットの種類に関係なく、フィールドの編集の可否や編集方法を制御するために設定できる共通のプロパティがあります。

ウィジェットの表示

ドラッグ＆ドロップ デザイナーモードでのみ利用でき、このグループはフィールドに割り当てられたウィジェットの外観を設定するのに役立ちます:

- **表示**: フィールド名をフォームに表示するかどうかを指定します
- **ラベルの色を上書き**: フィールドのラベルに特定の色を適用します
- **ラベルのフォントを上書き**: フィールドのラベルに特定のフォントプロパティ（太字、斜体、下線、取り消し線、フォントファミリー）を適用します

一般情報

- **別名（*Alias*）** ：フィールドに使用する、人間が読みやすい名前です。この別名は、地物フォームや属性テーブル、**地物情報** パネルに表示されます。また、これは **式文字列ビルダ** 内でのフィールド名の代わりとしても使用することができ、式の理解やレビューを用意にします。別名はプロジェクトファイル内に保存されます。
- **コメント** ： **フィールド** タブに表示されているフィールドのコメントを読み取り専用で表示します。この情報は地物フォームでフィールドのラベルにマウスカーソルを乗せた際のツールチップとして表示されます。
- **編集可能** ：このオプションのチェックを外すと、レイヤが編集モードであっても、このフィールドを読み取り専用（手動で変更不可）に設定します。この設定にチェックを入れても、プロバイダ側の編集制限は上書きできないことに注意してください。
- **最後に入力した値を利用する** ：このフィールドに最後に入力された値を記憶しておき、レイヤ内で次の地物を編集するときのデフォルト値として入力します。
- **ラベルを上に置く** ：地物フォームでフィールド名をウィジェットの上に置くか、横に置くかを設定します。

デフォルト値

- **デフォルト値** ：新しい地物に対して、デフォルトでフィールドを定義済みの値または **式ベースの値** で自動的に入力します。例を挙げると、以下のような値を定義できます:
  - `$x` や `$length` 、 `$area` を使用して、地物の作成時に地物の X 座標や長さ、面積やその他の幾何学的情報を自動的にフィールドに入力します。
  - `maximum("field")+1` を使用して、新しい地物を作成するたびにフィールドに 1 を加算します。
  - `now()` を使用して、地物の作成日時を保存します。
  - 式の中で **変数** を使用すると、作業者の名前（ `@user_full_name` ）やプロジェクトのファイルパス（ `@project_path` ）などを簡単に挿入できるようになります。

  結果のデフォルト値のプレビューがウィジェットの下部に表示されます。

注釈: デフォルト値 オプションは、作成される地物の他のフィールドの値を取得できないので、それらを組み合わせた式を使うことはできません。つまり、 `concat(field1, field2)` のような式は使えないということです。

- **更新時にデフォルト値を適用する** ：地物の属性やジオメトリに変更があった場合に、デフォルト値が再計算されます。このオプションは、データを変更した最後のユーザーや、最後に変更された時間などの値を保存するのに便利です。

制約

フィールドに入力する値を制限することができます。この制約には、以下の設定ができます：

- **非** *null*: ユーザーが値を入力することを必須とします。
- **ユニーク** ：入力された値がフィールド全体で一意であることを保証します。
- カスタム **式** に基づく制約：例えば、`not regexp_match(col0,'[^A-Za-z]')` は、*col0* フィールドの値にアルファベット文字しかないことを保証します。制約を覚えやすくするための短い説明文を追加することができます。

フィールドに値が追加されたり編集されるたびに、既存の制約条件が検証され、

- すべての要件を満たしている場合は、フォームのフィールドの横に緑色のチェック印が表示されます。
- 要件をすべて満たすことができていない場合には、フィールドは黄色またはオレンジ色で表示され、同じ色のバツ印がウィジェットの横に表示されます。バツ印にマウスカーソルを乗せると、どの制約条件が適用されいているのかを確認できるので、以下に従って値を修正します：
  - 満たされていない制約が強制のものでないとき（「ソフトな制約」）の場合、黄色いバツ印が表示され、「間違った」値で変更を保存することを妨げません；
  - オレンジ色のバツ印は無視することができず、制約を満たすまで変更を保存することができません。これは **制約を強制** オプションがチェックされているときに表示されます（「ハードな制約」）。

編集ウィジェット

フィールドの型に基づき、QGIS はデフォルトのウィジェットタイプを自動的に決定しフィールドに割り当てます。このウィジェットは、フィールドの型に適合する任意のウィジェットにユーザーが後から置き換えることが可能です。利用可能なウィジェットは以下のとおりです：

- **チェックボックス** ：チェックボックスを表示します。チェック状態で入力される値が決まります。
- **分類** ：レイヤに **カテゴリ値シンボロジ** が適用されている場合にのみ利用でき、クラス値のコンボボックスを表示します。
- **色** ： **カラーウィジェット** を表示し、色の選択ができます。色の値は html 表記で属性テーブルに保存されます。
- **日付/時刻** ：日付、時刻、またはその両方を入力するためのカレンダーウィジェットを開くことができる行フィールドを表示します。カラムの型はテキストでなければなりません。カスタム表示形式やカレンダーのポップアップなどの設定を選択することができます。
- **列挙** ：データベースから取得された既定の値のコンボボックスを開きます。これは現在のところ、PostgreSQL プロバイダで enum 型のフィールドでのみサポートしています。
- **アタッチメント** ：「ファイルを開く」ダイアログを使用して、ファイルパスを相対パスまたは絶対パスで保存します。これは、ハイパーリンク（ドキュメントへのパス）や画像、ウェブページ等を表示するために使用します。また、リソースを取得・保存するための **外部ストレージシステム** の設定もできます。

- **非表示** ：非表示の属性列は表示されません。ユーザーはフィールドの内容を見ることができません。
- **キー/値** 一つのフィールドにキー/値のペアのセットを保存するための 2 カラムのテーブルを表示します。これは現在のところ、PostgreSQL プロバイダで `hstore` 型のフィールドでのみサポートしています。
- **JSON を閲覧** ：シンタックスハイライトされた JSON データをテキストエディタまたはツリービューで表示します。このウィジェットは現在のところ、読み取り専用です。データがどのように表示されるかを変更するオプションがいくつかあります。「デフォルト表示形式」は、ウィジェットがテキストモードで表示されるか、ツリーモードで表示されるかを指定します。「JSON のフォーマット」には、ツリービューのみに関係する以下の 3 つのオプションがあります：
  - 字下げ：改行と 4 文字のスペースによるインデントを使用して、データを人間が読みやすい形式で表示します。
  - コンパクト：改行や空白を含まない、サイズ最適化された 1 行の文字列でデータを表示します。
  - 無効化：プロバイダからの形式どおりにデータを表示します。
- **リスト** ：一つのフィールドに様々な値を追加できる 1 カラムのテーブルを表示します。これは現在のところ、PostgreSQL プロバイダで `array` 型のフィールドでのみサポートしています。
- **範囲** ：特定の範囲から数値を設定できます。編集ウィジェットにはスライダーやスピンボックスもあります。
- **リレーションの参照** ：これは、 **リレーション** が設定されている場合に参照フィールド（つまりは子レイヤの外部キー）に指定されたフィールドのデフォルトのウィジェットです。 これにより親地物のフォームへ直接アクセスすることができ、その逆に親地物には子地物のリストとフォームが埋め込まれます。
- **テキスト編集** （デフォルト）：これは単純なテキストまたはマルチラインを使用できるテキスト編集フィールドを開きます。マルチラインを選択した場合は、HTML コンテンツを選択できます。
- **ユニーク値**：既に属性テーブルで使用されている値のいずれかを選択できます。「編集可能」が有効になっている場合、自動補完をサポートするラインエディットが表示されます。そうでない場合はコンボボックスが使用されます。
- **Uuid ジェネレータ** ：値が空の場合には読み取り専用の UUID（Universally Unique Identifiers）フィールドを生成します。
- **バリューマップ** ：事前に定義されたアイテムを持ったコンボボックスです。値は属性に格納され、コンボボックスに説明が表示されます。値は手動で定義するか、またはレイヤや CSV ファイルから読み込むことができます。
- **値のリレーション** ：関係テーブルの値をコンボボックスに表示します。レイヤ、キーカラム、値カラムを選択できます。標準的な動作を変更するために、NULL 値の許可、値による並べ替え、複数選択の許可、オートコンプリートの使用といったオプションが利用可能です。オートコンプリートチェックボックスを有効にすると、フォームにはドロップダウンリストまたはラインエディットフィールドが表示されます。

  レイヤが PostgreSQL、GeoPackage または SpatiaLite 形式で保存されており、 「値のリレーション」ウィジェットを使用する設定となっているが、必要となるレイヤがプロジェクトにまだ読み込まれていない場合、QGIS はそのレイヤを同じデータベース/接続から自動的に検索します。

Tip: アタッチメントウィジェットの相対パス

ファイルブラウザで選択されたパスが `.qgs` プロジェクトファイルと同じディレクトリかまたは下の階層に位置している場合、パスは相対パスに変換されます。これにより、 `.qgs` プロジェクトにマルチメディア情報がアタッチメントされている場合の可搬性が向上します。

16.1.10 テーブル結合プロパティ

 **テーブル結合** タブでは、現在のレイヤ（ ターゲットレイヤ と呼びます）の地物と他に読み込んだベクタレイヤの地物（またはテーブル）を関連付けることができます。結合は、レイヤ間で共有されている属性値に基づいて行われます。レイヤはジオメトリなし（テーブル）でも、ジオメトリありでも良いですが、 結合する属性は同じ型である必要があります。

テーブル結合を作成するには：

1.  新しい結合を追加 ボタンをクリックします。 **ベクタ結合を追加** ダイアログが現れます。
2. ターゲットベクタレイヤと結合させたい **結合するレイヤ** を選択します。
3. 結合レイヤとターゲットレイヤで共通の、 **結合基準の属性** と **ターゲット属性** を指定します。
4. *OK* ボタンを押すと、選択したパラメータの概要が **テーブル結合** パネルに表示されます。

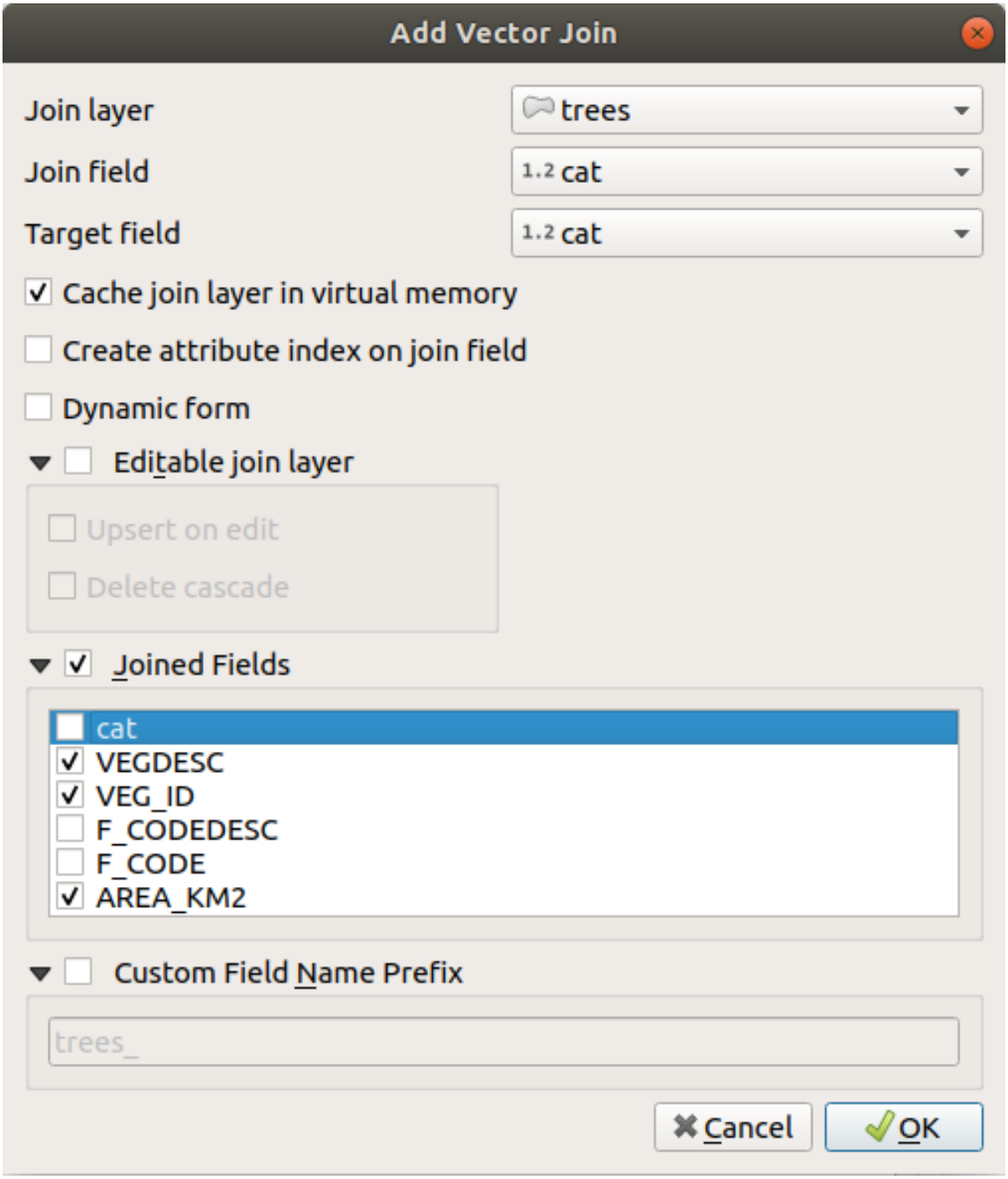


図 16.44: 既存のベクタレイヤに属性テーブルを結合する

上で述べたステップでテーブル結合が作成されますが、結合レイヤの最初にマッチした地物の **全ての** 属性値がターゲットレイヤの地物に追加されます。QGIS には、結合を微調整するためのオプションがさらにあります：

- **結合レイヤをキャッシュ** ：ルックアップを高速化するために、結合レイヤの値を（ジオメトリなしで）メモリにキャッシュできるようになります。
- **結合属性にインデックスを作成**
- **動的フォーム（結合レイヤと連動）** ： **ターゲットフィールド** の変更に応じて結合フィールドを

オンザフライで同期できるようになります。これにより、結合フィールドの制約条件も適切に更新されます。多数の地物があったり無数の結合がある場合には非常に時間がかかることがあるため、この機能はデフォルトでは無効化されていることに注意してください。

- ターゲットレイヤが編集可能な場合には、フィールドの横の属性テーブルにはステータスを知らせるためのアイコンが表示されます：
  -  ：結合レイヤが編集可能に設定されていません。ターゲットの属性テーブルから結合地物を編集できるようにしたい場合は、  **編集可能な結合レイヤ** オプションにチェックを入れる必要があります。
  -  ：結合レイヤは編集可能に設定されていますが、現在のステータスは読み取り専用です。
  -  ：結合レイヤは編集可能ですが、同期メカニズムは有効ではありません。ターゲットレイヤに地物が作成されたときに、結合レイヤに地物を自動的に追加したい場合には、  **編集時** *Upsert* オプションにチェックを入れる必要があります。これとは反対に、結合した地物を自動的に削除したい場合には、  **カスケード削除** を有効にする必要があります。
-  **結合フィールド** ：結合するレイヤの全てのフィールドを追加するのではなく、その一部分を指定することができます。
-  **カスタムフィールド名の接頭辞** ：名前の衝突を回避するために付けられる結合フィールド名の接頭辞をカスタマイズできます。

QGIS は現在のところ、GDAL でサポートされている非空間テーブル（例えば CSV、DBF、Excel など）や区切りテキスト、PostgreSQL プロバイダに対する結合をサポートしています。

16.1.11 補助テーブルプロパティ

スタイルやラベルをカスタマイズするには、**データによって定義された上書きの設定** で説明しているように、データによって定義されたプロパティを使用するのが一般的です。しかし、元となるデータが読み取り専用の場合には、これができない場合があります。さらに、データ定義のプロパティの設定では非常に時間がかかり、望ましくない場合さえもあります。例えば、**ラベルツールバー** にあるマップツールを完全に使用したい場合には、元のデータソースに 20 以上のフィールド（X・Y 位置、回転角度、フォント、色など）を追加して設定する必要があります。

補助テーブルのメカニズムは、これらの制限や扱いにくい設定に対する解決策です。補助フィールドは、これらのデータ定義のプロパティ（ラベル、ダイアグラム、シンボロジなど）を自動的に管理し、編集可能なテーブル結合によって SQLite データベース内に保存するための間接的な方法です。 これにより、編集可能ではないレイヤのプロパティを保存することができます。

ベクタレイヤプロパティダイアログには、補助テーブルを管理するためのタブがあります。

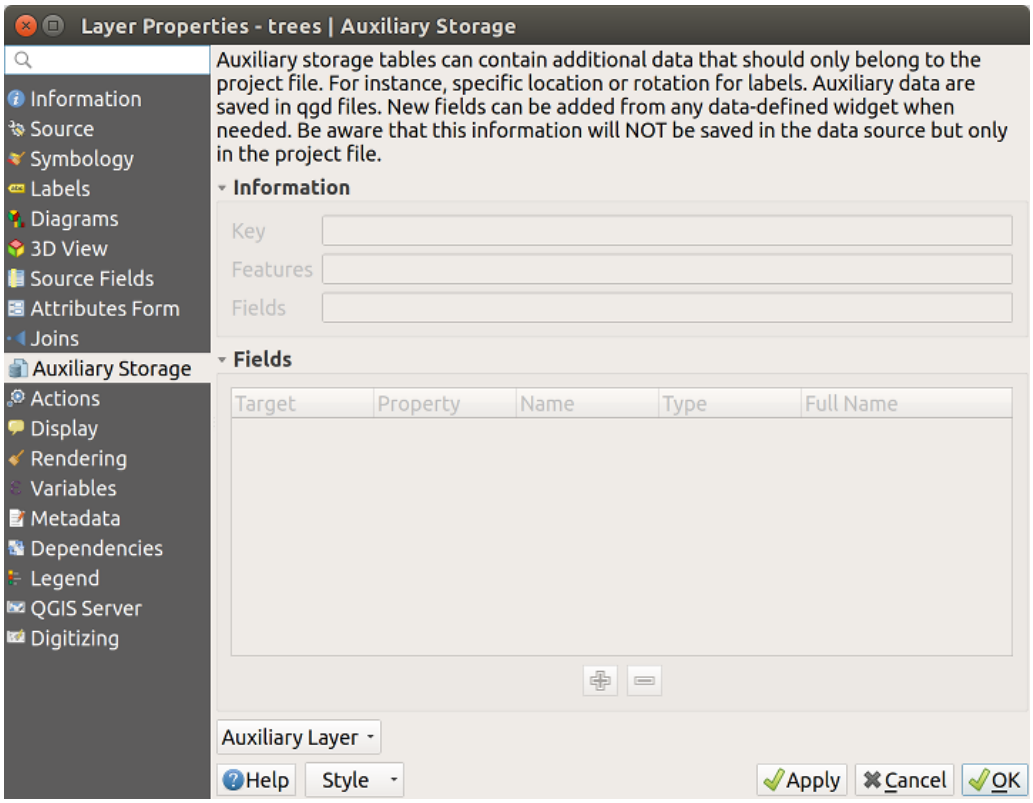


図 16.45: 補助テーブルタブ

ラベル

編集可能でなくともデータソースはデータ定義のプロパティによってカスタマイズできることを考慮すると、 **ラベルツールバー** で説明しているラベリングマップツールは、ラベリングが有効になると同時に常に利用可能になります。

実際のところ、補助テーブルシステムはこれらのプロパティを SQLite データベース（ **補助テーブルのデータベース** 参照）に保存するための補助レイヤが必要です。ラベリングマップツールがアクティブになっている状態で初めてマップをクリックしたときに、補助レイヤの作成プロセスが実行されます。このときウィンドウが表示され、結合に使用する主キーを選択します（地物が一意に識別されることを保証するため）。

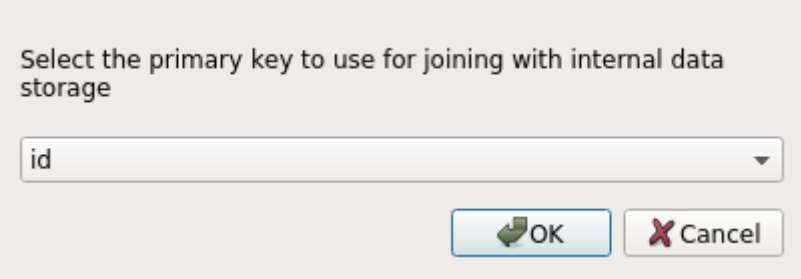


図 16.46: 補助レイヤの作成ダイアログ

現在のデータソースに補助レイヤが設定されると、補助テーブルタブでその情報を検索することができます：

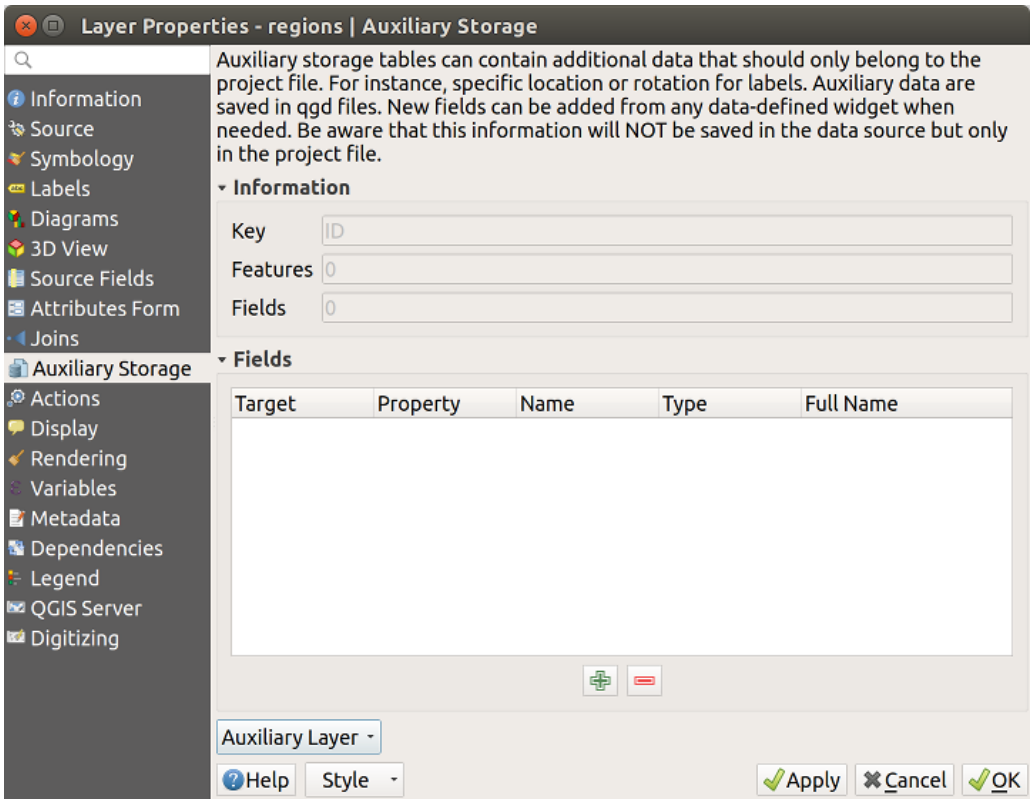


図 16.47: 補助レイヤのキー

補助レイヤには次のような特徴があります：

- 主キーは `ID`
- 補助属性を使用した `0` 個の地物がある
- `0` 個の補助属性がある

補助レイヤが作成されたので、レイヤのラベルを編集することができます。 ラベル属性の変更 マップツールが有効な状態でラベルをクリックすると、サイズや色などのスタイリングプロパティを更新することができます。これに対応したデータ定義のプロパティが作成され、この情報は検索することができます。

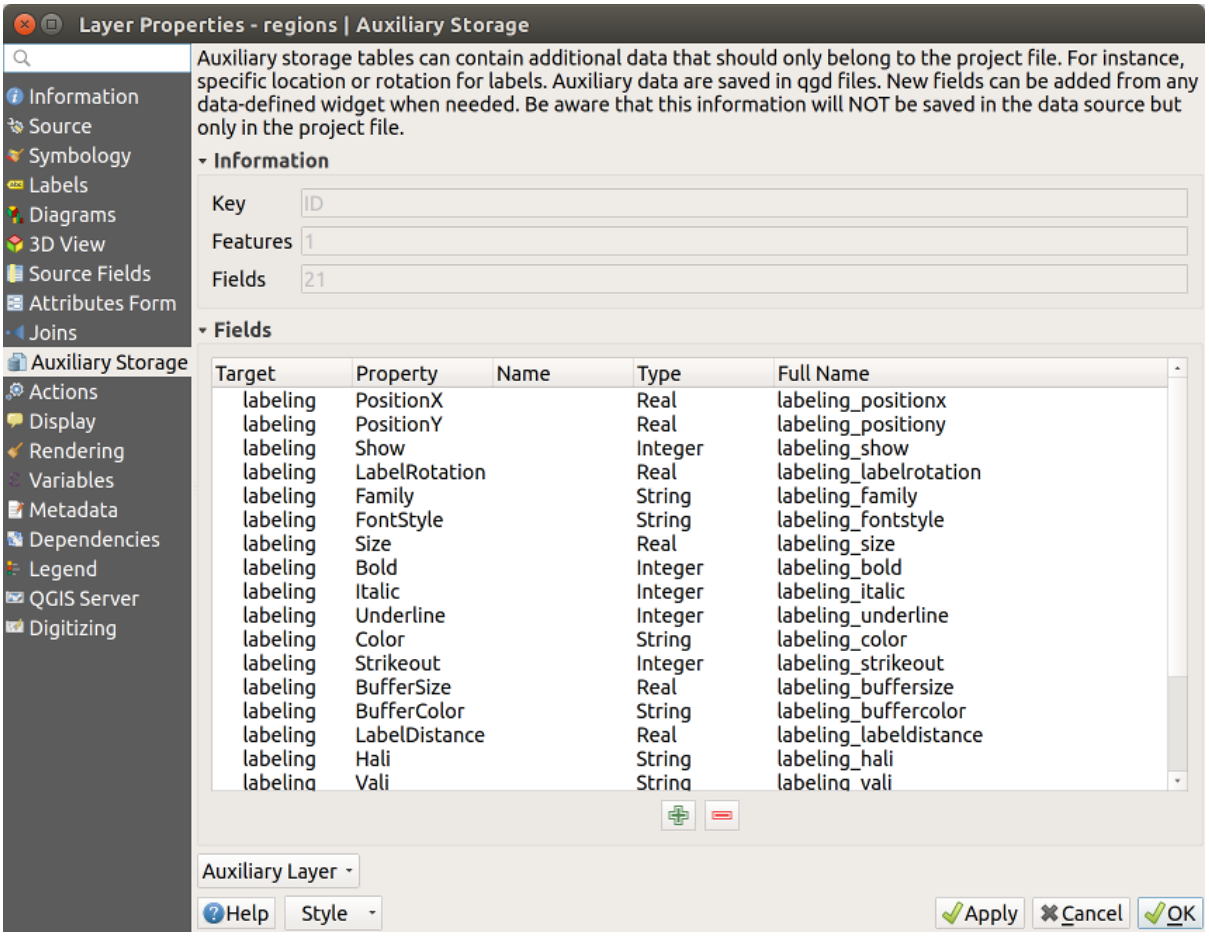


図 16.48: 補助属性

上図でわかるように、 21 個の属性が自動的に作成され、ラベリング用に設定されています。例えば、`FontStyle` 補助属性の型は `String` であり、元の SQLite データベース内では `labeling_fontstyle` という名前が付けられています。これらの補助属性を使用している地物が 1 つあることもわかります。

ラベル プロパティタブ内で  アイコンが表示されていることに注目してください。これは、データ定義の上書きオプションが正しく設定されていることを示しています。

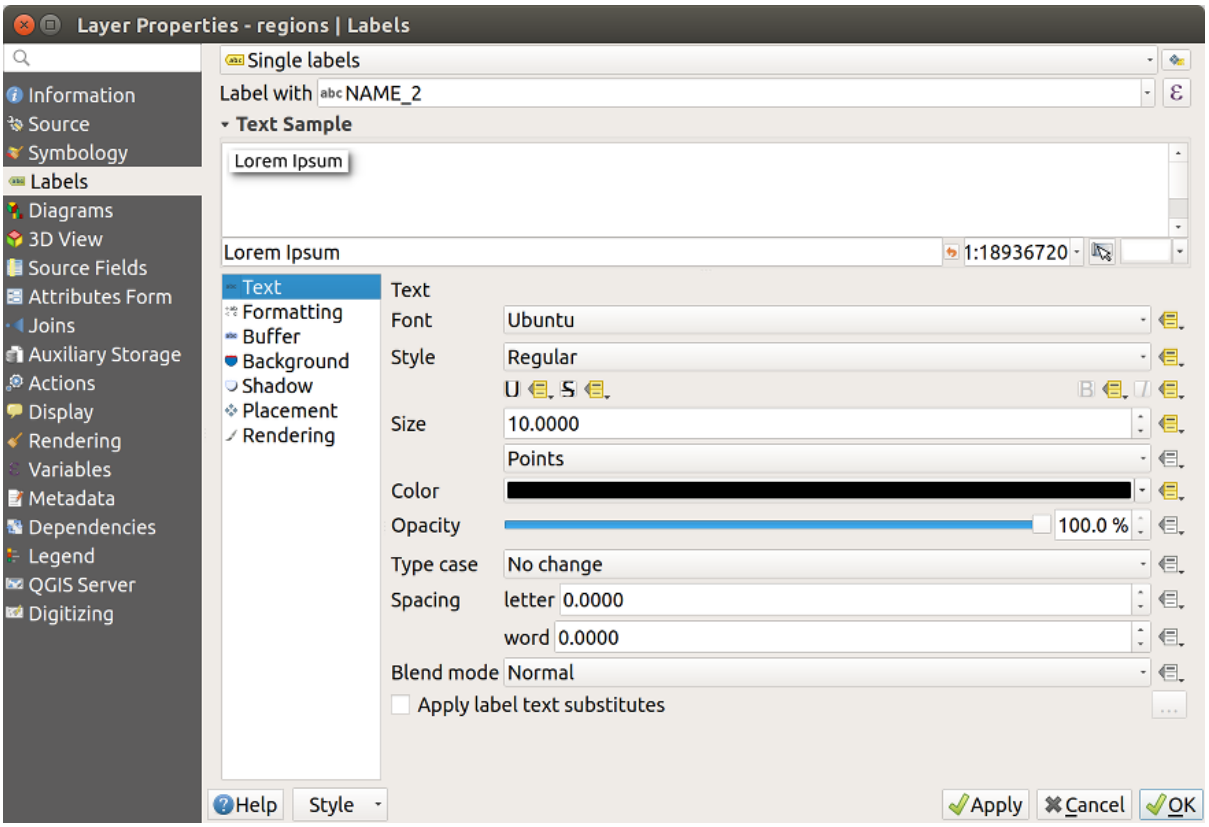


図 16.49: データ定義のプロパティの自動生成

この方法の他に、 データによって定義された上書き ボタンを使用して特定のプロパティのための補助属性を作成する方法もあります。 **データをプロジェクトに格納する** をクリックすると、例えば **不透明度** フィールド用の補助属性が自動的に作成されます。補助レイヤがまだ作成されていない状態でこのボタンをクリックすると、まずは結合に使用する主キーを選択するためのウィンドウ（ 図 16.46 ）が表示されます。

シンボロジ

上で述べたラベルのカスタマイズ方法と同じように、補助属性を使ってシンボルやダイアグラムのスタイル設定ができます。これを行うには、 データによって定義された上書き をクリックして、特定のプロパティに対して **データをプロジェクトに格納する** を選択します。例えば、 **塗りつぶし色** 属性の場合には以下の図のようになります。

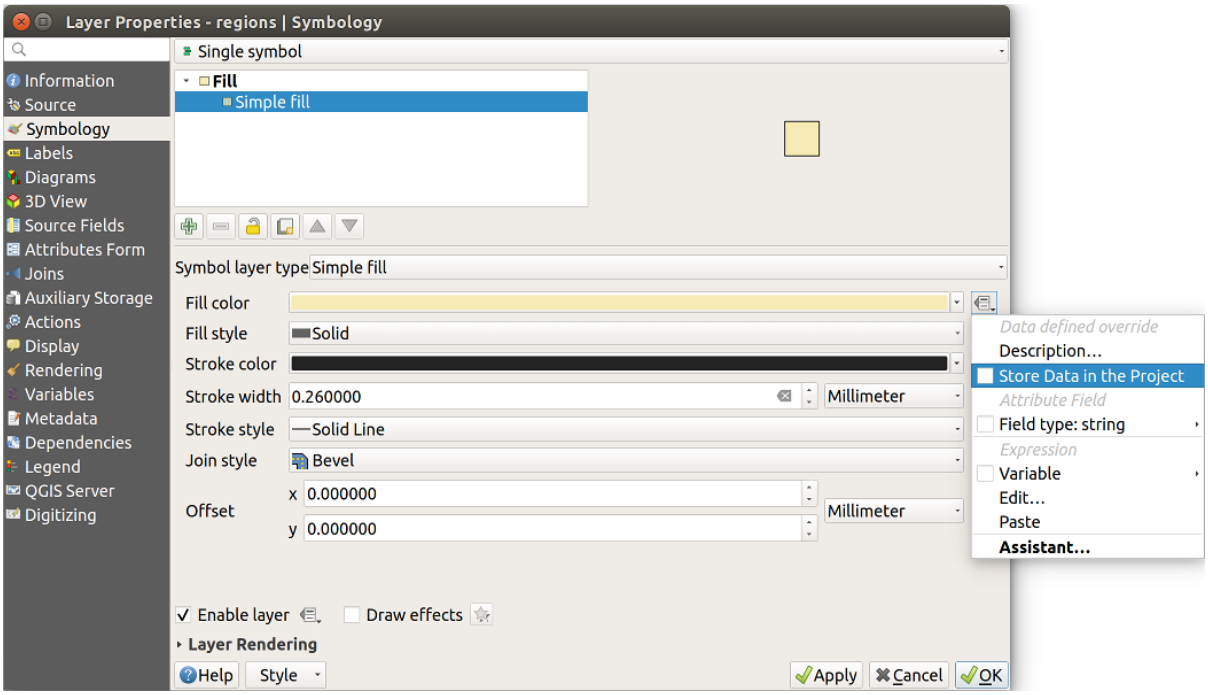


図 16.50: シンボルに対するデータ定義のプロパティのメニュー

各シンボルにはさまざま属性（塗りつぶしスタイル、塗りつぶし色、ストローク色など）があるため、属性を表現する各補助属性には名前の衝突を避けるためにユニークな名前が必要です。 **データをプロジェクトに格納する** を選択すると、ウィンドウが開き属性の **型** が表示され、補助属性のユニークな名前を入力するよう求められます。例えば **塗りつぶし色** の補助属性を作成すると、次のようなウィンドウが開きます。

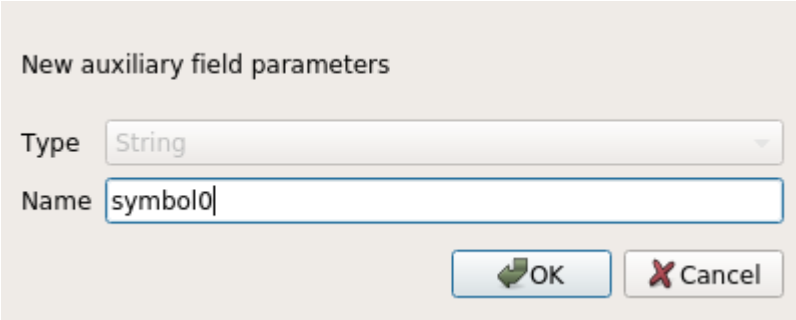


図 16.51: シンボルの補助属性の名前

作成できたら、この補助属性は補助テーブルタブから検索することが可能になります。

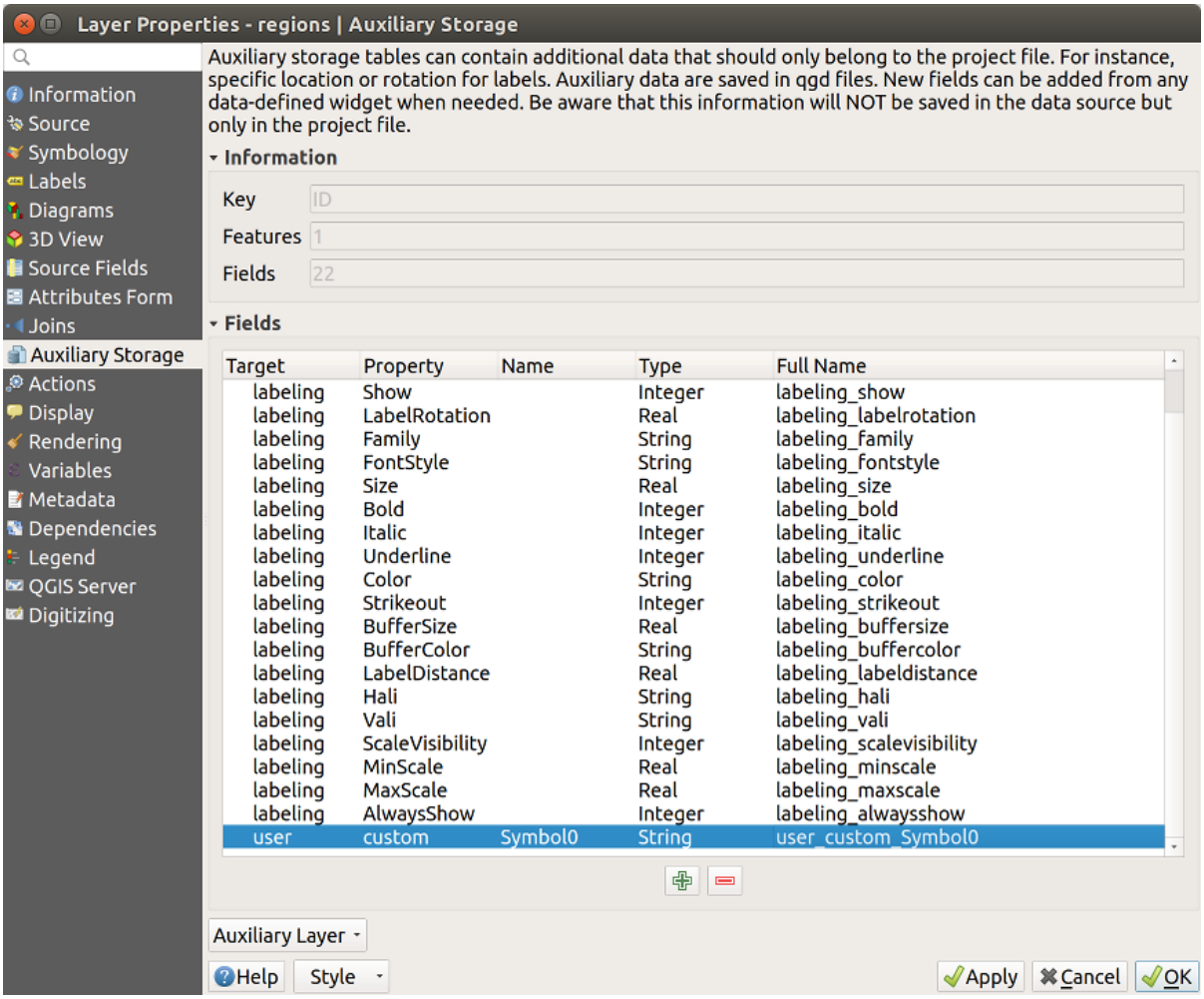


図 16.52: シンボルの補助属性

属性テーブルとウィジェット

補助属性は **属性テーブル** を使用して編集することができます。ただし、全ての補助属性が最初から属性テーブルに表示されているわけではありません。

補助属性のうち、レイヤのシンボロジやラベリング、表示方法やダイアグラムなどの属性を表現するものは、自動的に属性テーブルに表示されます。例外は **ラベルツールバー** を使用して修正可能な属性で、これはデフォルトでは非表示となっています。 色 を表す補助属性には **色** ウィジェットがデフォルトで設定され、これ以外の補助属性のデフォルトは **テキスト編集** ウィジェットです。

ラベルツールバー を使用して修正可能な補助属性は、属性テーブルではデフォルトで **非表示** になっています。属性を表示するためには **属性フォームプロパティタブ** を開き、補助属性の **ウィジェットタイプ** の値を **非表示** から他の関連する値に変更します。例えば、 **auxiliary_storage_labeling_size** を **テキスト編集** ウィジェットへと変更したり、 **auxiliary_storage_labeling_color** を **色** ウィジェットに変更したりする等です。そうすると、これらの属性は属性テーブルに表示されるようになります。

属性テーブル内での補助属性は以下の画像のように表示されます。

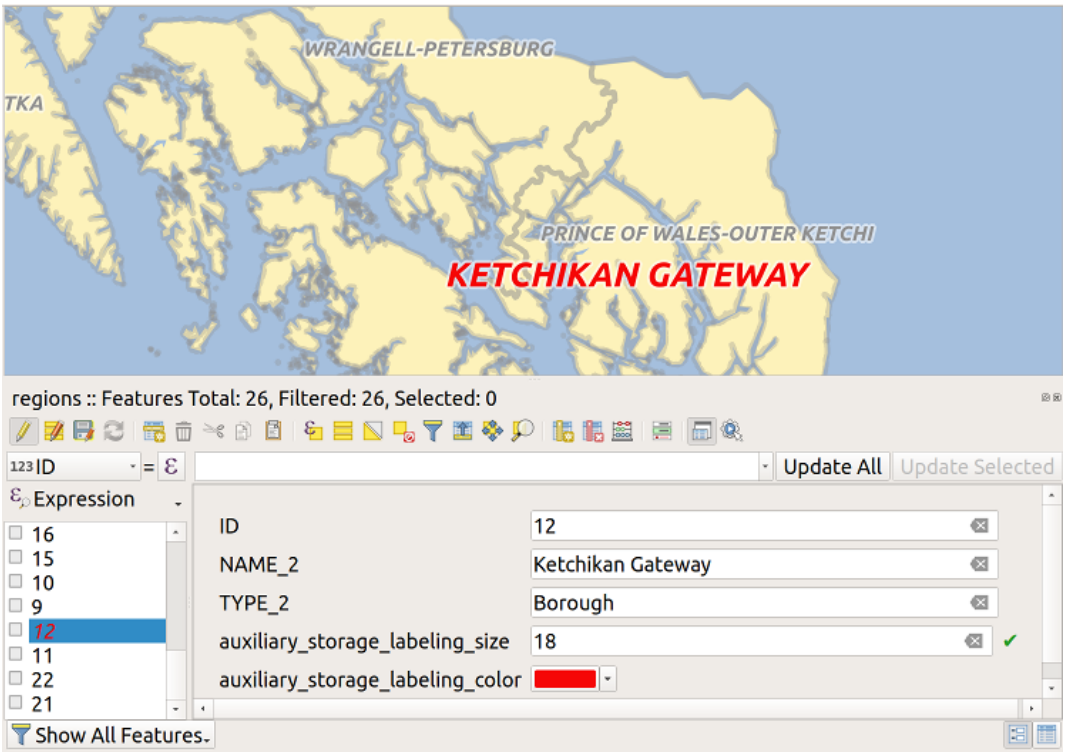


図 16.53: 補助属性のフォーム

管理

補助テーブル メニューでは、補助属性の管理ができます。

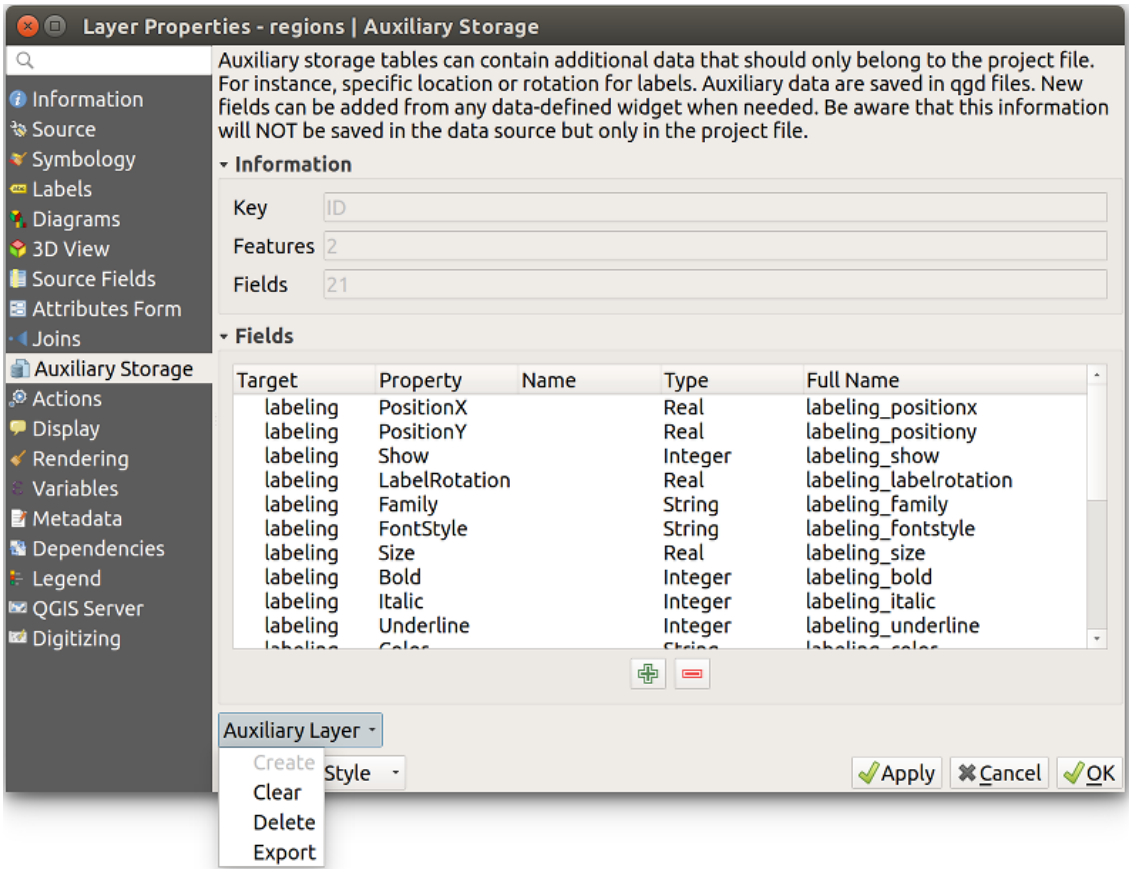


図 16.54: 補助テーブルの管理

最初のアイテム **作成する** は、補助テーブルがすでに作成されているため、この画像では無効化されています。新たな作業の場合は、このアクションを使って補助テーブルを作成することができます。 **ラベル** で説明しているように、この時には主キーが必要です。

クリア アクションは、全ての補助属性を残しますが、その値を削除します。これにより、これらの属性を使用している地物の数は `0` になります。

削除 アクションは補助テーブルを完全に削除します。言い換えると、対応するテーブルは元の SQLite データベースから削除され、プロパティのカスタム設定が失われます。

最後の **エクスポート** アクションでは、補助テーブルを **新しいベクタレイヤ** として保存できます。ジオメトリは補助テーブルには保存されていないことに注意してください。ただし、この時にジオメトリをもとのデータソースから一緒にエクスポートすることはできます。

補助テーブルのデータベース

プロジェクトを `.qgs` 形式で保存する場合には、補助テーブルに使用している SQLite データベースはプロジェクトと同じ場所に拡張子 `.qgd` で保存されます。

この代わりに `.qgz` 形式によるアーカイブを使用するのが利便性のためには良いでしょう。この場合には `.qgd` ファイルと `.qgs` ファイルは両方ともアーカイブ内に埋め込まれます。

16.1.12 アクションプロパティ

QGIS には地物の属性に基づいてアクションを実行する機能があります。これは任意の数のアクションを実行するために使用できます。例えば、地物の属性から構築された引数でのプログラムの実行や、Web レポーティングツールにパラメータを渡す、などです。

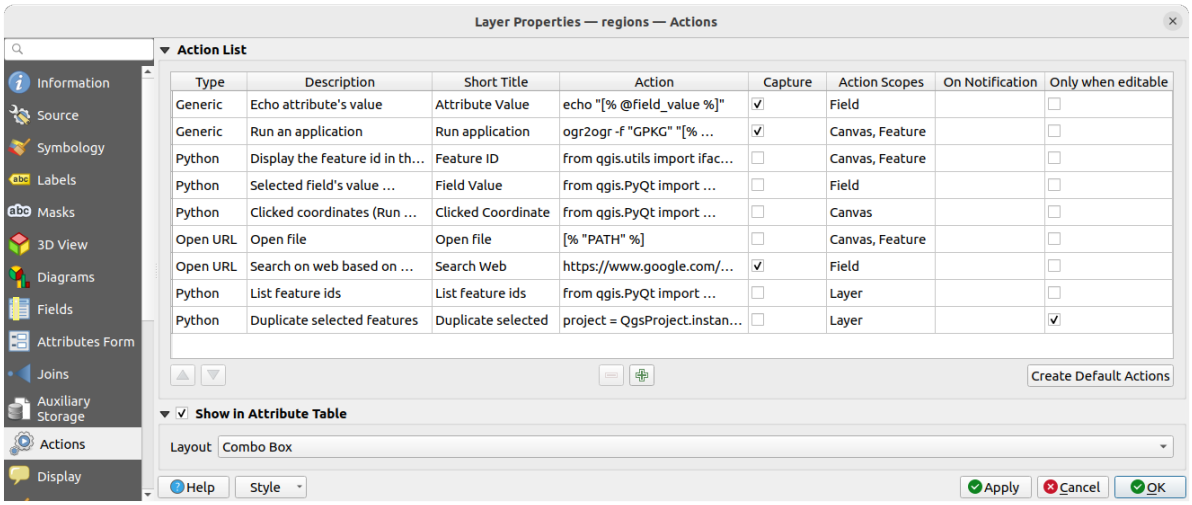


図 16.55: いくつかのサンプルアクションを含むアクションダイアログの概要

アクションが便利なのは、頻繁に外部のアプリケーションを実行したり、ベクタレイヤの 1 つ以上の値に基づいて Web ページを表示したりする場合です。それらには様々な種類があり、次のように使うことができます:

- **一般**、 *macOS*、 *Windows*、 *Unix* アクションは外部プロセスを開始します。
- *Python* アクションは Python コードを実行します。
- **一般** と *Python* アクションは、どのプラットフォームでも表示されます。
- *macOS*、*Windows*、*Unix* アクションは、それぞれのプラットフォーム上だけに表示されます（つまり、エディタを開くための「編集」アクションは 3 つ定義することができ、ユーザーはプラットフォームに応じたただ 1 つの「編集」アクションのみが表示され、そのアクションを実行してエディタを開きます）。
- *URL* **を開く**: 与えられた URL を開く HTTP GET リクエストを使います。

- *URL* **送信** *(urlencoded/JSON)*： *URL* **を開く** アクションと同じですが、HTTP POST リクエストを使います。データは、"application/x-www-form-urlencoded"、またはボディが有効な JSON の場合は "application/json "を使って、URL にポストされます。

  アクションの呼び出しの例は次のようなものになります:

  ```
  http://localhost:8000?/[% url_encode(map('file', 'index.php')) %]
  ```

- *URL* **送信** *(multipart)*：*URL* **を開く** アクションと同じですが、HTTP POST リクエストを使います。データは"multipart/form-data"を使って URL に投稿されます。

ダイアログにはいくつかの例が含まれています。 **デフォルトアクションを作成** ボタンをクリックするとそれらを読み込むことができます。例のどれかを編集するには、行をダブルクリックします。例の一つに属性値に基づいた検索の実行があります。これは以下の議論で使用します。

属性テーブルに表示 にチェックを入れると、地物スコープにチェックの入ったアクションを **コンボボックス** または **別々のボタン** として属性テーブルダイアログ内に表示させることができます（ **列の設定** を参照）。

アクションの定義

属性のアクションを定義するには、ベクタ **レイヤプロパティ** ダイアログを開いて **アクション** タブをクリックしてください。 **アクション** タブ内で アクションを追加 をクリックすると、 **アクションの編集** ダイアログが開きます。

アクションの **型** を選択し、アクションにわかりやすい名前を与えます。アクション自体には、アクションが呼び出されたときに実行されるアプリケーションの名前が含まれている必要があります。アプリケーションへの引数として、1 つ以上の属性フィールドの値を追加できます。アクションが呼び出されると、 `%` で始まるフィールド名の文字列は、そのフィールドの値に置き換えられます。特別な文字列 `%%` は、地物情報表示結果や属性テーブルから選択されたフィールドの値に置き換えられます（**アクションを使用する** を参照）。二重引用符は、テキストをプログラムやスクリプト、コマンドの単一引数にまとめるために使用することができます。バックスラッシュが前置された二重引用符は無視されます。

アクションの **スコープ** では、どこでアクションが有効となるかを定義することができます。次の選択肢があります：

1. **フィールド** ：アクションは属性テーブル内のセルで右クリックした場合や地物フォーム内、あるいはメインツールバーのデフォルトアクションボタンで利用可能です。
2. **地物** ：アクションは属性テーブル内のセルで右クリックした場合に利用可能です。
3. **キャンバス** ：アクションはツールバーのメインアクションボタンで利用可能です。
4. **フォーム**: アクションは **ドラッグ＆ドロップ** モードを使用してデザインされた地物フォームでのみ利用可能です。
5. **レイヤ** ：アクションは属性テーブルのツールバー内のアクションボタンで利用可能です。この型のアクションは、単一の地物ではなくレイヤ全体を対象としていることに注意してください。

フィールド名が他のフィールド名の部分文字列である場合（例えば `col1` と `col10` ）、フィールド名（と `%` 文字）を角括弧で囲むことで示す必要があります（例えば `[%col10]` ）。 こうすることで `%col10` フィール

ド名が、末尾に 0 がある %col1 フィールド名と間違えることを防げます。角括弧は QGIS がフィールドの値を代入する際に除去されます。置換されたフィールドを角括弧で囲みたい場合は、次のように 2 つ目の組を使用します：[[%col10]]

地物情報表示 ツールを使用すると、**地物情報** ダイアログを開くことができます。これには、レイヤのタイプに関連した情報が含まれる **（派生した属性）** アイテムを含みます。このアイテムの値は派生したフィールド名の頭に (Derived). を付けることで、他のフィールドと同じ方法でアクセスできます。例えばポイントレイヤには X と Y というフィールドがあり、これらのフィールドの値はアクション内で、%(Derived).X と %(Derived).Y で使用できます。派生した属性は **地物情報** ダイアログボックスからのみ利用可能で、**属性テーブル** ダイアログボックスからは利用できません。

以下に 2 つのアクション例を示します：

- konqueror https://www.google.com/search?q=%nam
- konqueror https://www.google.com/search?q=%%

最初の例では、ウェブブラウザ konqueror を起動して開くべき URL を渡しています。この URL は、ベクタレイヤの nam フィールドの値の Google 検索を実行します。アクションによって呼び出されるアプリケーションやスクリプトはパスが通っているか、もしくはフルパスで指定する必要があることに注意してください。念のため、最初の例を次のように書き換えてみましょう：/opt/kde3/bin/konqueror https://www.google.com/search?q=%nam 。これにより、アクションが呼ばれたときには konqueror アプリケーションが確実に実行されるようになります。

2 つ目の例では %% 表記を使用しています。この値は特定のフィールドに依存しません。アクションが呼び出されると、 %% は地物情報や属性テーブルの選択したフィールドの値に置き換えられます。

アクションを使用する

QGIS には、レイヤに対して有効にしたアクションを実行するための多数の方法があります。設定に応じて、次のような方法が利用できます：

- **属性ツールバー** や **属性テーブル** ダイアログの  地物アクションの実行 ボタンのドロップダウンメニュー
-  地物情報表示 ツールで地物を右クリック（更なる情報は **地物の識別** 参照）
- **地物情報** パネルの **アクション** セクション
- **属性テーブル** ダイアログ内の **アクション** 列のアイテム

%% 表記を使用するアクションを起動する場合には、**地物情報** ダイアログや **属性テーブル** ダイアログ内で、アプリケーションやスクリプトに渡したいフィールド値を右クリックしてください。

ここでは、bash と echo コマンドを使って、ベクタレイヤからデータを取り出してファイルに挿入する別の例を紹介します（従って  またはおそらく  でしか動作しません）。本件のレイヤには、種名 taxon_name 、緯度 lat 、経度 long のフィールドがあります。ここでは、地域を空間的に選択すると、（QGIS マップエリアで黄色に表示されている）選択されたレコードのフィールド値をテキストファイルにエクスポートできるようにしたいと思います。これを達成するためのアクションは次のとおりです。

```
bash -c "echo \"%taxon_name %lat %long\" >> /tmp/species_localities.txt"
```

いくつかの地域を選択してそれぞれでアクションを実行し、出力ファイルを開くと、以下のようなものが表示されます。

```
Acacia mearnsii -34.0800000000 150.0800000000
Acacia mearnsii -34.9000000000 150.1200000000
Acacia mearnsii -35.2200000000 149.9300000000
Acacia mearnsii -32.2700000000 150.4100000000
```

練習問題として、 lakes レイヤで Google 検索を実行するアクションを作成してみましょう。まずは、あるキーワードで検索を実行するために必要な URL を見つけ出す必要があります。これは、Google にアクセスして簡単な検索を行い、ブラウザのアドレスバーから URL を取得するだけで簡単にできます。少しの手間で、 QGIS が検索語のときには URL の形式は https://www.google.com/search?q=QGIS であることがわかりました。この情報をもとに次へと進みましょう。

1. lakes レイヤが読み込まれていることを確認してください。
2. 凡例内のレイヤをダブルクリックして **レイヤプロパティ** ダイアログを開くか、またはレイヤを右クリックしてポップアップメニューから **プロパティ** を選択します。
3. **アクション** タブをクリックします。
4.  アクションを追加 ボタンをクリックします。
5. アクション型に *URL* **を開く** を選択し、
6. アクションの名前を入力します。例えば Google 検索 など。
7. さらに **短いタイトル** や **アイコン** も追加することができます。
8. アクションの **スコープ** を選択します。詳細については **アクションの定義** を参照してください。この例ではデフォルトの設定のままとします。
9. アクションには、Google 検索に使用する URL を検索語を含まないところまで追加します：`https://www.google.com/search?q=`
10. ここまでで **アクション** フィールドのテキストは次のようになっています:

    ```
    https://www.google.com/search?q=
    ```

11. lakes レイヤのフィールド名を含むドロップダウンボックスをクリックします。これは **挿入** ボタンのすぐ左にあります。
12. ドロップダウンボックスから、 :guilabel:'NAMES' を選択して **挿入** をクリックします。
13. するとアクションテキストは次のようになります:

    ```
    https://www.google.com/search?q=[%NAMES%]
    ```

14. アクション作成を終了して追加するために、 *OK* ボタンをクリックします。

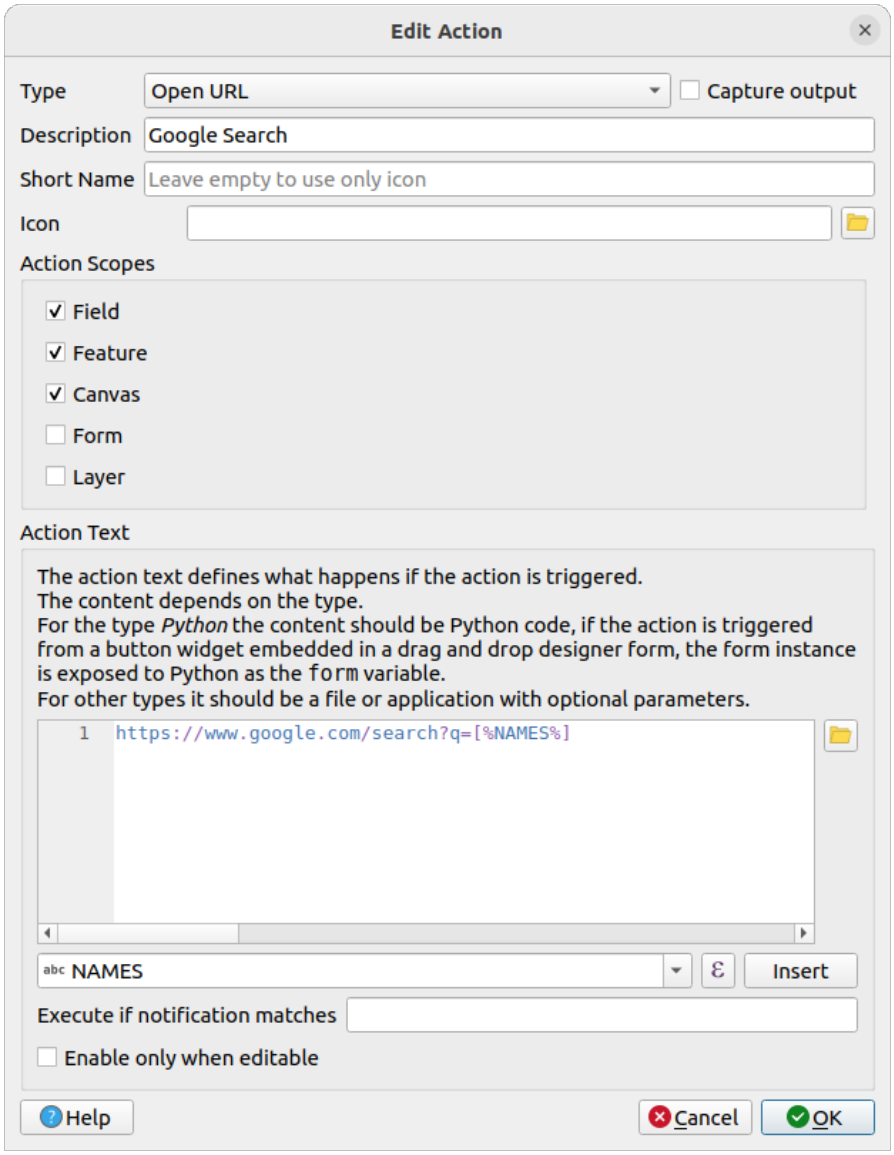


図 16.56: この例で設定されたアクションの編集ダイアログ

これでアクションが完成し、使用できるようになりました。

レイヤプロパティ ダイアログを閉じ、地図を見たい領域にズームして下さい。 `lakes` レイヤがアクティブであることを確認して、地物情報表示ツールで湖をクリックして下さい。地物情報ボックスの中にアクションが表示されているはずです：

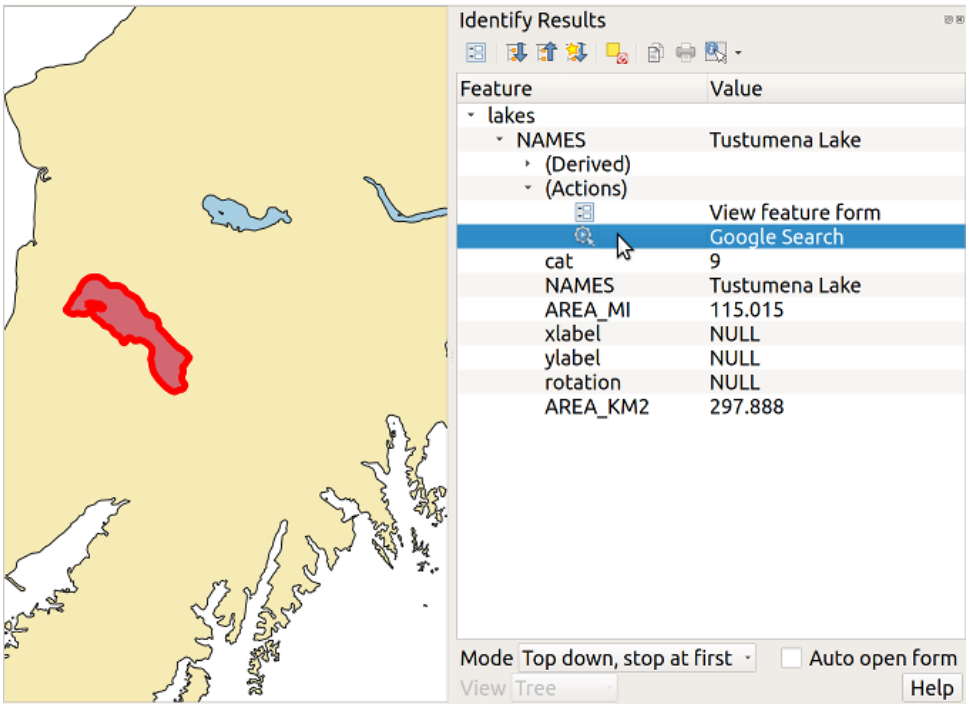


図 16.57: 地物を選択し、アクションを選択する状態

アクションをクリックすると、デフォルトのブラウザが立ち上がり、 URL https://www.google.com/search?q=Tustumena に移動します。アクションにさらに属性フィールドを追加することも可能です。これにはアクションテキストの末尾に + を追加し、別のフィールドを選択して **挿入** をクリックします。ただしこの例では、検索する意味があるような他のフィールドは持っていません。

レイヤには複数のアクションを定義することができ、 **地物情報** ダイアログにはそれぞれが表示されます。

また、属性テーブルからアクションを呼び出すには、属性テーブルの行を選択して右クリックし、ポップアップメニューからアクションを選択します。

アクションにはさまざまな使い方があります。例えば、画像や写真の位置とファイル名を含むポイントレイヤがあれば、画像を表示するビューアを起動するアクションを作成することができます。また、アクションを使って、Google 検索の例と同じように属性フィールドやフィールドの組み合わせを指定して、web ベースのレポートを起動することもできるでしょう。

また、より複雑な例を、例えば **Python** アクションを使用して作成できます。

通常、外部アプリケーションでファイルを開くためのアクションを作成する際には絶対パスまたは相対パスを使用することができます。後者のケースでは、パスは外部プログラムの実行ファイル位置に対して相対的になります。しかし、選択したレイヤ（シェープファイルや SpatiaLite のようなファイルベースのもの）に対する相対パスを使用する必要がある場合にはどうでしょうか？次のようなコードで対応することができます。

```
command = "firefox"
imagerelpath = "images_test/test_image.jpg"
layer = qgis.utils.iface.activeLayer()
import os.path
layerpath = layer.source() if layer.providerType() == 'ogr'
```

(次のページに続く)

(前のページからの続き)

```
  else (qgis.core.QgsDataSourceURI(layer.source()).database()
    if layer.providerType() == 'spatialite' else None)
path = os.path.dirname(str(layerpath))
image = os.path.join(path,imagerelpath)
import subprocess
subprocess.Popen( [command, image ] )
```

このアクションは *Python* 型であり、 *command* と *imagerelpath* 変数は必要に応じて変更する必要があることは覚えておく必要があります。

では、相対パスが（保存された）プロジェクトファイルに相対的である必要がある場合はどうでしょうか？ Python アクションのコードは以下のようになります。

```
command = "firefox"
imagerelpath = "images_test/test_image.jpg"
projectpath = qgis.core.QgsProject.instance().fileName()
import os.path
path = os.path.dirname(str(projectpath)) if projectpath != '' else None
image = os.path.join(path, imagerelpath)
import subprocess
subprocess.Popen( [command, image ] )
```

もう一つの Python アクションの例は、プロジェクトに新しいレイヤを追加することができるものです。例えば、以下の例では、ベクタレイヤとラスタレイヤそれぞれをプロジェクトに追加します。プロジェクトに追加されるファイルの名前やレイヤに付けられる名前はデータ駆動型です（ *filename* と *layername* は、アクションが作成されたベクタレイヤの属性テーブルの列名です）。

```
qgis.utils.iface.addVectorLayer('/yourpath/[% "filename" %].shp',
  '[% "layername" %]', 'ogr')
```

ラスタ（この例では TIF 画像）を追加するには、次のようになります：

```
qgis.utils.iface.addRasterLayer('/yourpath/[% "filename" %].tif',
  '[% "layername" %]')
```

16.1.13 表示名プロパティ

表示名 タブでは、地物の識別に使用するフィールドの設定ができます。

- **表示名** ：フィールドまたは **式** に基づきます。これは以下のように利用されます：
  - **地物情報表示ツール** の結果の地物情報の先頭に表示されるラベル
  - **ロケータバー** で全てのレイヤから地物を検索する際に使用されるフィールド
  - 属性テーブルの **フォームビュー** での地物の識別子

  - マップやレイアウトを GeoPDF 等のレイヤ出力形式にエクスポートする際の地物識別子
  - 地図の Tips 情報、すなわち、 

アイコンを押した状態でアクティブレイヤの地物上にマウスカーソルを重ねた際に、マップキャンバスに表示されるメッセージ。 **地図の *Tips* の *HTML* 定義** が何も設定されていない場合に適用されます。

- **地図の *Tips* の *HTML* 定義** は、地図の Tips のために特別に作成するものです。これはフィールドや式、HTML タグ（マルチライン、フォント、画像、ハイパーリンク等）が混在した、より複雑で完全な HTML テキストです。

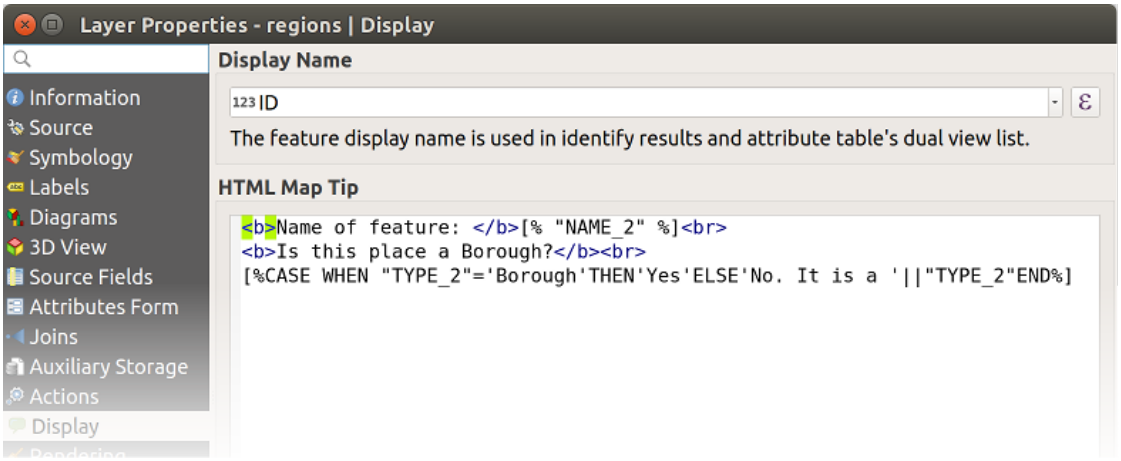


図 16.58: 地図の Tips の HTML コード

地図の Tips を有効にするためには、 **ビュー　地図の *Tips* を表示** メニューオプションを選択するか、 **属性ツールバー** の 

アイコンをクリックします。地図の Tips はセッション横断的な機能で、一度有効にすると無効にするまでは有効な状態が継続し、どのプロジェクトのどのレイヤにも、次の QGIS セッションでさえも適用されます。

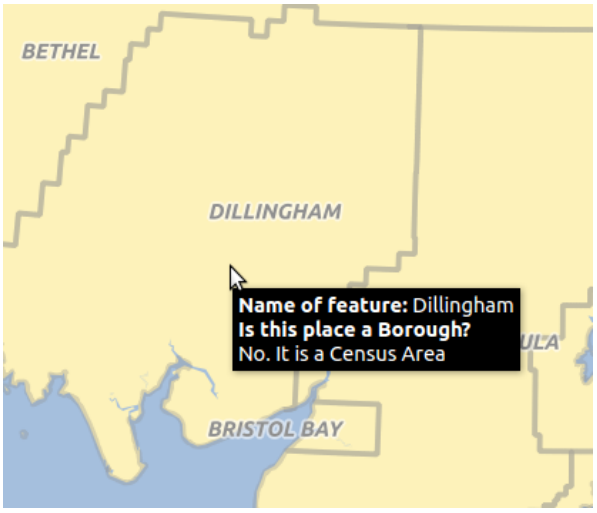


図 16.59: HTML コードで作られた地図の Tips

16.1.14 レンダリングプロパティ

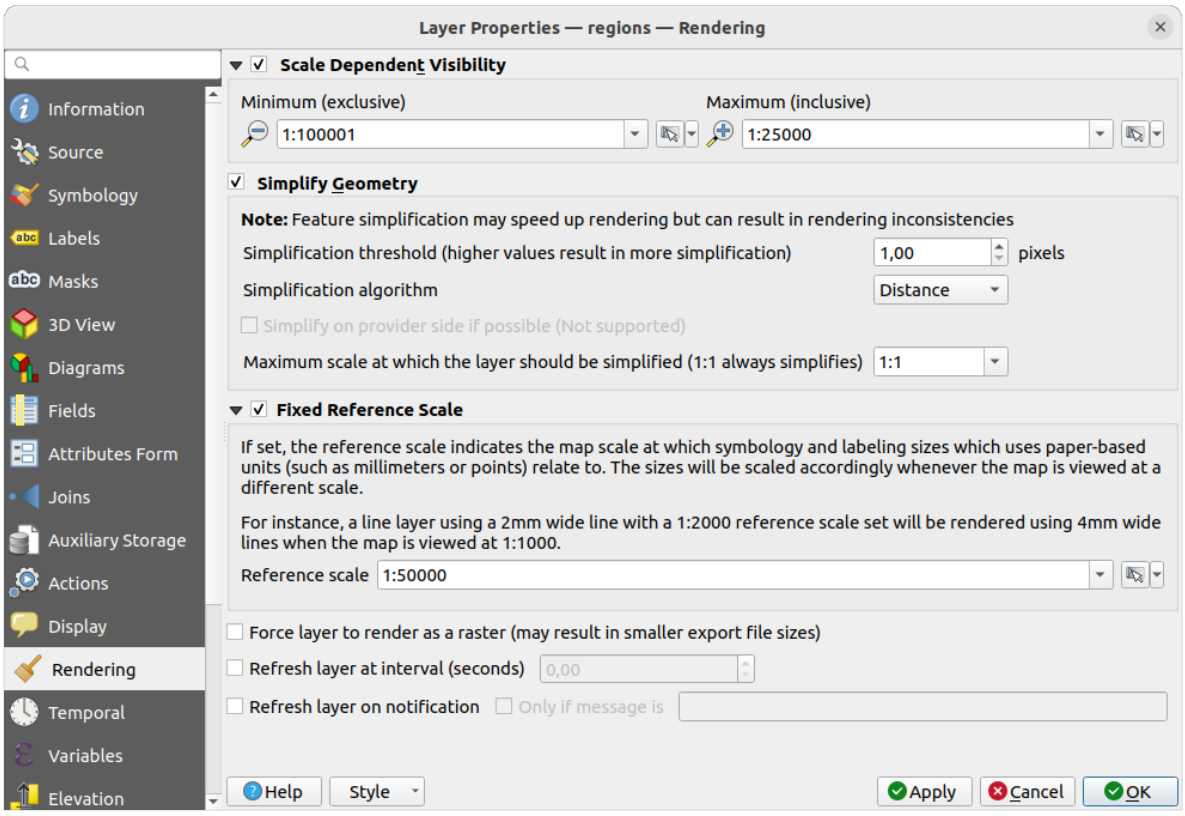


図 16.60: レイヤのレンダリングプロパティダイアログ

縮尺に応じた表示設定

最大（含む） と **最小（含まない）** 縮尺を設定して、地物が表示される縮尺の範囲を定義できます。この範囲の外では地物は非表示になります。 現在のキャンバスの縮尺に設定 ボタンを使用すると、可視性の範囲の境界として現在のマップキャンバスの縮尺を使用できます。詳細については **表示縮尺セレクタ** 参照してください。

注釈: レイヤの縮尺に応じた表示は、**レイヤ** パネルからも有効にすることもできます: レイヤを右クリックし、コンテキストメニューから **レイヤの縮尺表示を設定** を選択します。

ジオメトリを簡素化

QGIS はオンザフライでの地物の簡素化をサポートしています。これにより、小縮尺で多数の複雑な地物を描画する際のレンダリング時間を短縮することができます。この機能は、レイヤの設定の **ジオメトリの簡素化** オプションで有効または無効にすることができます。また、新しく追加されたレイヤに対して、デフォルトで簡素化を有効にするグローバル設定もあります（詳細は **グローバルな地物の簡素化** を参照してください）。

注釈: 地物の簡素化により、場合によってはレンダリングされた出力に不整合が発生することがあります。これには、ポリゴン間の細長い隙間の発生や、オフセットに基づくシンボルレイヤの不正確なレンダリングなどがあります。

非常に詳細なレイヤ（膨大な数のノードを持つポリゴンレイヤなど）をレンダリングすると、すべてのノードをエクスポートファイルに含めるため、PDF/SVG 形式でのレイアウトエクスポートが巨大になることがあります。これによって、この結果ファイルを他のプログラムで作業したり開いたりする際に非常に遅くなることもあります。

レイヤをラスタとして描画する にチェックを入れると、これらのレイヤがラスタ化されるので、このようなレイヤに含まれるノード全てをエクスポートするファイルに含める必要がなくなり、レンダリングが高速化されます。

レイアウトを強制的にラスタとして書き出すことでも対応できますが、これは全てのレイヤにラスタ化が適用される、オール・オア・ナッシングの解決策です。

再描画間隔（秒） ：タイマーを設定して、マッチした間隔で個々のレイヤを自動的に更新します。複数のレイヤに自動更新間隔が設定されている場合、複数回の更新を避けるためにキャンバスの更新は延期されます。

データプロバイダによっては、QGIS の外でデータソースに変更が適用されたときに、QGIS へ通知を送信することができます（例：PostgreSQL）。 **通知時にレイヤを更新する** オプションを使用すると、通知によって更新を実行できます。また、 **メッセージがある場合のみ** のテキストボックスで設定した特定のメッセージの場合にのみレイヤを更新するよう制限することもできます。

参照スケールの使用

これを設定すると、紙ベースの単位（ミリメートルやポイント等）を使用するシンボルやラベルのサイズに関係する地図スケールについての参照スケールとなります。地図が異なる縮尺で表示された場合、サイズはこれに応じて拡大縮小されます。

例えば、1:2000 の参照スケールが設定されている 2mm 幅のラインレイヤは、1:1000 で表示した場合には 4mm 幅のラインでレンダリングされます。

16.1.15 時系列プロパティ

時系列 タブは、時間経過に伴うレイヤのレンダリングを制御するオプションを提供します。このような動的なレンダリングには、マップキャンバスで **時系列ナビ** を有効にする必要があります。

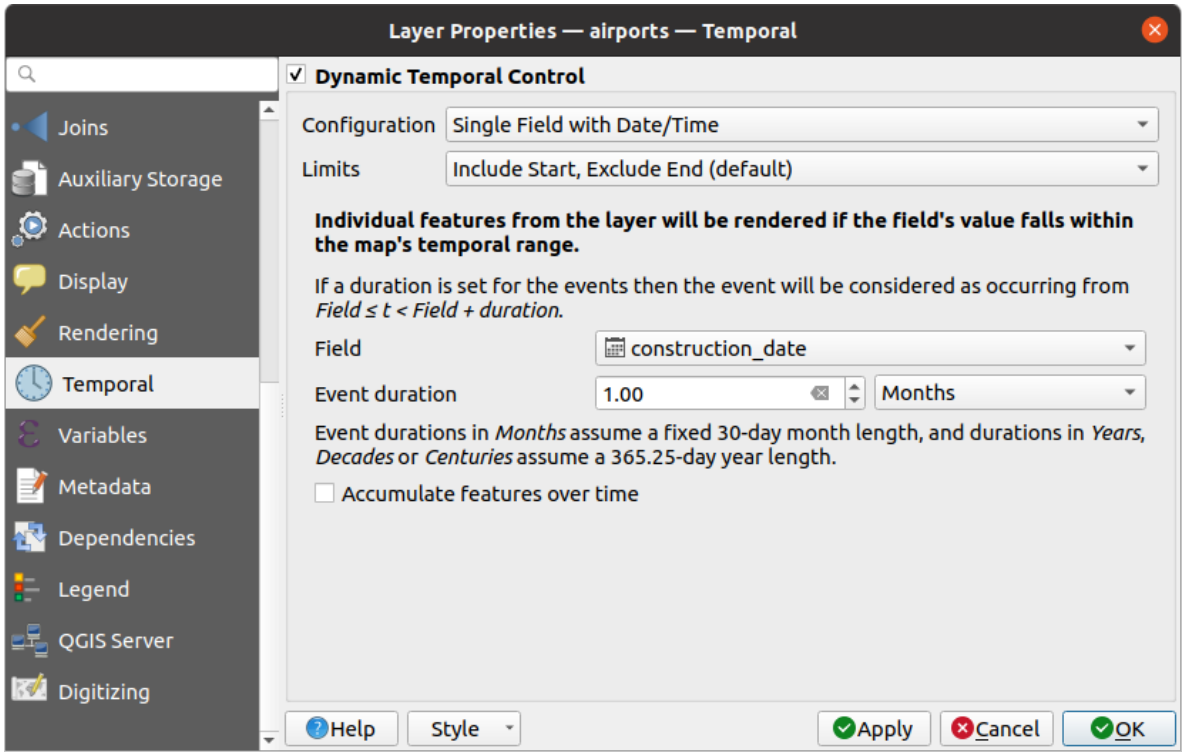


図 16.61: ベクタレイヤの時系列プロパティダイアログ

ベクタレイヤの時系列レンダリングを設定するには、 **動的時系列コントロール** オプションにチェックを入れます。データセットの構造にもよりますが、 **設定** オプションのいずれかを使用します：

- **固定時間範囲** ：マップキャンバスの時間フレームが指定する **開始時刻** と **終了時刻** の範囲と重なる場合に、すべての地物がレンダリングされます。
- *Date/Time* **型の単一フィールド** ：地物の **フィールド** の値がマップキャンバスの時間フレーム内にある場合に、地物がレンダリングされます。 **イベントの継続時間** を設定することができます。 **地物を累積表示** オプションにチェックを入れると、マップの時間フレーム内かそれ以前に発生したすべての地物をレンダリングし続けます。従ってイベントの継続時間は無視されます。
- **開始・終了の日時のフィールド** ： **開始フィールド** と **終了フィールド** の値で指定される範囲がマップキャンバスの時間フレームと重なる場合に地物がレンダリングされます。
- **開始・終了の継続時間のフィールド** ： **開始フィールド** と **継続時間フィールド** の値で定義される範囲がマップキャンバスの時間フレームと重なる場合に地物がレンダリングされます。
- **式による開始・終了日時** ：フィールドの **開始式** と **終了式** で指定される時間範囲がマップキャンバスの時間フレームと重なる場合に地物がレンダリングされます。
- **レイヤのみ再描画** ：レイヤは新しいアニメーションフレーム毎に再描画されますが、地物に対する時間ベースのフィルタリングは何も適用されません。これは、レンダラーの設定に時間ベースの式の値を使用しているレイヤ（例：データによって定義されたシンボロジ）に便利です。

また、地物の時間範囲の **上限** を次のように設定することが可能です：

- **始点を含み終点を除く**
- **始点も終点も含む**

16.1.16 変数プロパティ

変数 タブでは、レイヤのレベルで利用可能なすべての変数（すべてのグローバルおよびプロジェクトの変数も含む）を示しています。

また、ここではユーザーはレイヤレベルの変数の管理もできます。 ボタンをクリックして、新しいレイヤレベルのカスタム変数を追加します。同様に、リストからレイヤレベルのカスタム変数を選択して ボタンを押すと、変数が削除されます。

一般ツールの **値を変数に格納する** セクションには、変数の使用に関する詳細があります。

16.1.17 標高プロパティ

標高 タブには、 *3D* **マップビュー** 内のレイヤの標高プロパティと、 *profile tool charts* 内のレイヤの標高をコントロールするオプションがあります。具体的には次が設定できます:

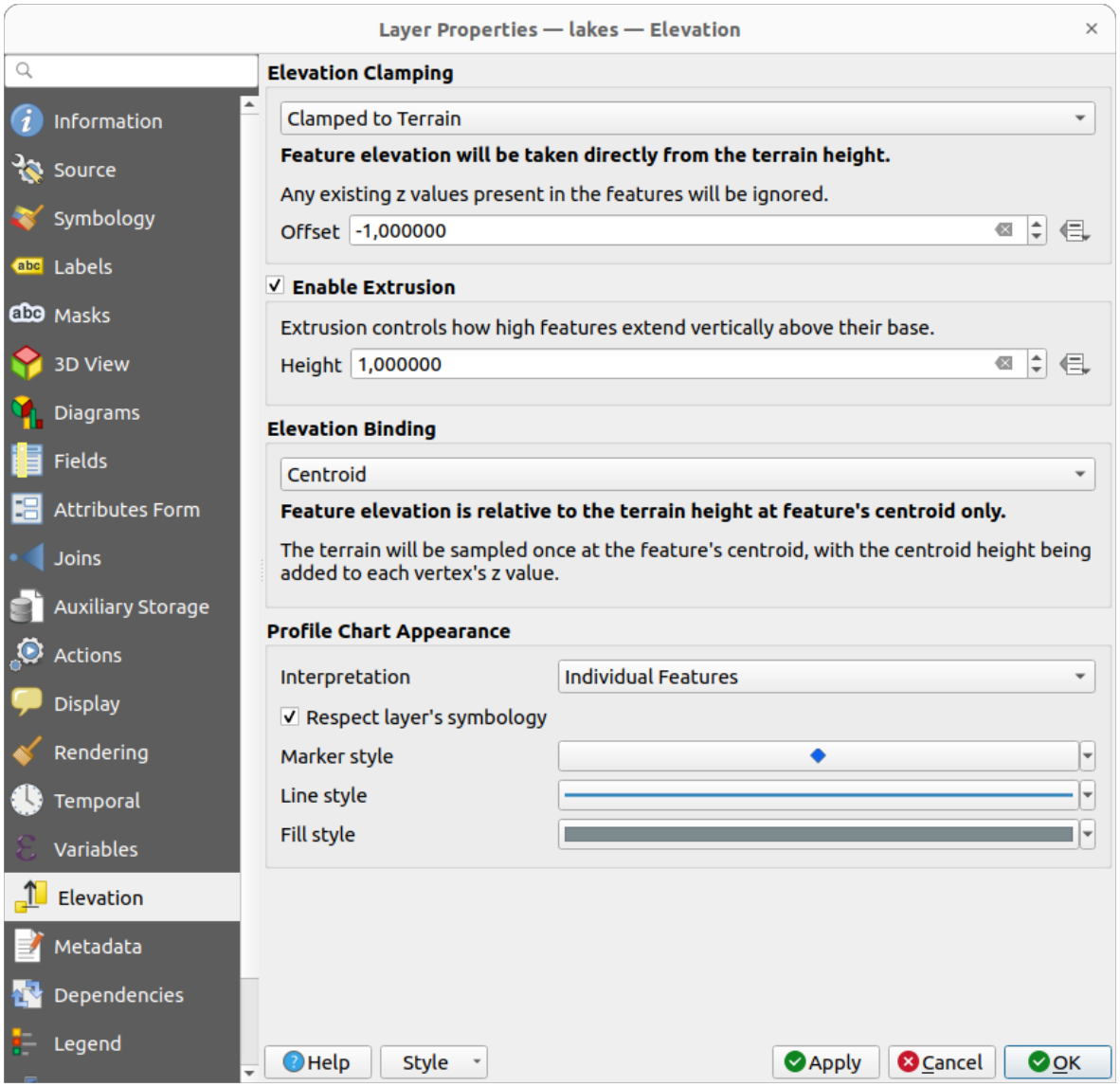


図 16.62: ベクタレイヤの標高プロパティダイアログ

- **標高クランピング**: 地物の高度をどのように、またどちらにするかを定義します：
  - **地形の高さ**: 地形の高さから直接標高を取得し、地物内の既存の Z 値を無視します。データ定義の地形からの **オフセット** 値を入力することもできます。
  - **地形に相対的**: 地物にある Z 値が地形の高さに加算されます。**スケール** 係数とデータ定義の **オフセット** を使って標高を調整することができます。このオプションは 2D ジオメトリレイヤでは利用できません。
  - **地物の絶対位置**: 地形の高さを無視し、地物の Z 値を直接標高に使います。**スケール** 係数とデータ定義の **オフセット** を使って標高を調整することができます。2D ジオメトリレイヤ（Z 値を持たない）の場合、代わりにデータ定義の **基準高** を設定することができます。
- **押し出しを有効化**: **高さ** を設定することで、地物がベースから垂直方向にどれだけ高く伸びるかを制御することができます。これは、2D ジオメトリレイヤ、例えばポリゴンの建物の足跡レイヤが、実際には 3D オブジェクトを表していることを示すのに便利です。

- **標高バインディング**: **標高クランピング** に依存する地形をラインレイヤまたはポリゴンレイヤと組み合わせる場合にのみ関係するオプションで、地形の高さに対する地物の標高の設定方法を制御します。地形は次の位置からサンプリングすることができます:
  - 地物の **重心** では、各頂点の z 値に重心の高さが加えられます
  - 個々の **頂点** では、その頂点の z 値に加えられる前に
- **断面図の外観**: 断面図を描画する際に、地物をどのように表示するかを制御します。主に 2 つの **解釈** モードが利用できます：
  - **各地物**: 断面図線がベクタ地物と交差する離散的な位置をサンプリングします。この交差はレイヤの種類や押し出しの有無によって、点、線、面として表現されます。

    レイヤのシンボルに従う をチェックすると、地物は対応する **レイヤのスタイル** で断面図上にレンダリングされます（例えば、カテゴリ値の分類が断面図上で見えるようになります）。断面図のシンボルの型がレイヤのレンダラのシンボルの型と一致しない場合、レンダラのシンボルの色のみが断面図シンボルに適用されます。

    レイヤの設定によっては、断面図のシンボルをカスタム・スタイルで表現することができます：

    * **マーカースタイル**: 押し出しのないポイント地物とライン地物、および断面図線に接する押し出しのないポリゴン地物用
    * :ref:`線のスタイル `: 押し出しのあるポイント地物とライン地物、および断面図線と交差する押し出しのないポリゴン地物用
    * **塗りつぶしスタイル**: 押し出しのあるポリゴン地物用
  - **連続的サーフェス（等高線など）**: 標高チャートは、サンプリングされた標高の結果を連続した線に結合し、別々の地物ではなくサーフェスとしてレンダリングされます。これは視覚化を向上させることができ、等高線や測量された標高点など、連続した標高面を表すベクタレイヤ用に設計されています。線の **スタイル** は次のように設定できます：
    * **線のスタイル** が適用された、断面 **線**
    * 対応する:guilabel:**塗りつぶしスタイル** `で *:guilabel:* `**下を塗りつぶす** されたサーフェス

    さらに、 **マーカーをサンプリング表示** をチェックすることで、判読線上にマーカーを表示し、それらに **マーカースタイル** を割り当てることができます。

16.1.18 メタデータプロパティ

メタデータ タブには、レイヤに関するメタデータレポートの作成・編集オプションがあります。詳細は **メタデータ** を参照してください。

16.1.19 依存関係プロパティ

依存関係 タブでは、レイヤ間のデータの依存関係を宣言することができます。データの依存性は、あるレイヤのデータ修正が、ユーザーの直接操作が無くとも他のレイヤのデータを修正してしまう場合に発生します。これには例えば、あるレイヤのジオメトリが、他のレイヤのジオメトリを修正した後に、データベーストリガーやカスタム PyQGIS スクリプトによって更新される場合などがあります。

依存関係 タブでは、現在のレイヤのデータを外部から変更する可能性のあるレイヤを任意で選択できます。依存するレイヤを正しく指定すれば、依存するレイヤが変更されたときに、QGIS はこのレイヤのキャッシュを無効化することができます。

16.1.20 凡例プロパティ

凡例 プロパティタブでは、**レイヤパネル** や **印刷レイアウトの凡例** に関する高度な設定ができます。これらのオプションには以下のものがあります：

- レイヤに適用しているシンボロジにもよりますが、凡例に複数のエントリが表示されることで、必ずしも読みやすく便利な表示になるとは限りません。 **凡例の枠の画像 `で代わりとなる** *:ref:* **`画像の選択** *<embedded_file_selector>* を行うことができ、 **レイヤ** パネルや印刷レイアウトの凡例の両者で表示されます。
- **ラベル凡例を表示** ：さまざまなラベル設定の概要を凡例内のエントリとして表示します。 **ラベルのスタイル** が、説明とともにプレビューされます。
- **シンボル上のテキスト** ：場合によっては、凡例内のシンボルに追加の情報を加えると便利なことがあります。このフレームでは、レイヤのシンボロジで使用されているシンボルのどれにでもテキストつけることができます。このテキストは、 **レイヤ** パネルとプリントレイアウトの凡例の両方でシンボル上に表示されます。テーブルウィジェットでシンボルの横に各テキストを入力するか、 **式でラベルを設定** ボタンを使ってテーブルに入力することで、凡例とテキストの対応付けがなされます。テキストの外観は **テキスト形式** ボタンのフォント・色セレクタウィジェットで設定します。

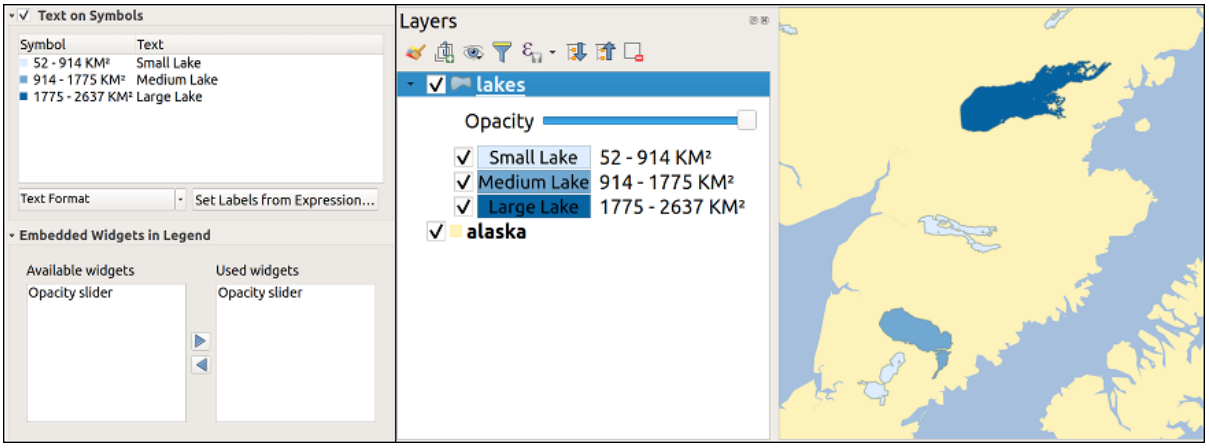


図 16.63: シンボル上のテキスト設定（左）と **レイヤ** パネルでのレンダリング結果（右）

- レイヤパネル内のレイヤツリーに埋め込み可能なウィジェットのリスト。このアイディアは、レイヤで頻繁に使用するいくつかのアクション（不透明度の設定、フィルタ、選択、スタイルなど）に簡単にアクセスするための方法を提供することです。

デフォルトでは QGIS は不透明度ウィジェットを提供していますが、これは独自ウィジェットを登録するプラグインによって拡張することが可能で、管理するレイヤにカスタムアクションを割り当てられます。

16.1.21 QGIS サーバープロパティ

QGIS **サーバー** タブは、 **説明** 、 **帰属** 、 **メタデータ** *URL* および **凡例** *URL* のセクションからなります。

説明 セクションでは、リクエスト内でのレイヤ参照に使用される **短い名前** （短い名前の詳細については、services_basics_short_name を参照）を変更できます。また、レイヤの **タイトル** や **要約** を追加・編集したり、 **キーワードリスト** をここに定義することができます。このキーワードリストはメタデータカタログで使用することができます。タイトルを XML メタデータファイルから使用したい場合には、 **データ** *URL* フィールドにリンク先を入力する必要があります。

帰属 を使用して XML メタデータカタログから属性データを取得します。

メタデータ *URL* では、XML メタデータカタログへの一般的なパスを追加できます。この情報はその後のセッションのために QGIS プロジェクトファイル内に保存され、QGIS サーバーで使用されます。

凡例 *URL* セクションでは、凡例画像の URL を URL フィールドに指定します。「形式」ドロップダウンメニューのオプションを使用して、適切な画像形式を選択できます。現在、PNG および JPG・JPEG 画像形式をサポートしています。

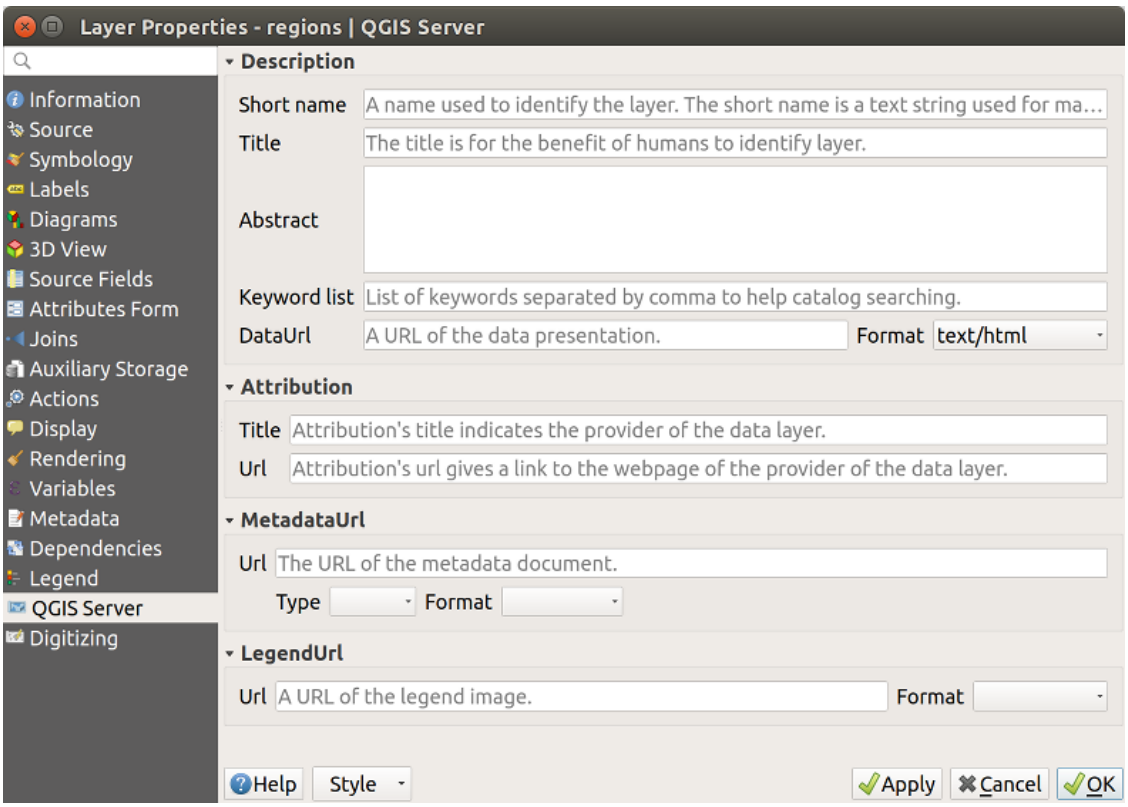


図 16.64: ベクタレイヤプロパティダイアログの QGIS サーバータブ

QGIS サーバーについて更に学ぶには、 QGIS-Server-manual を読んでください。

16.1.22 デジタイズプロパティ

デジタイズ タブにはデジタイズしたジオメトリの品質を確保するために役立つオプションへのアクセスがあります。

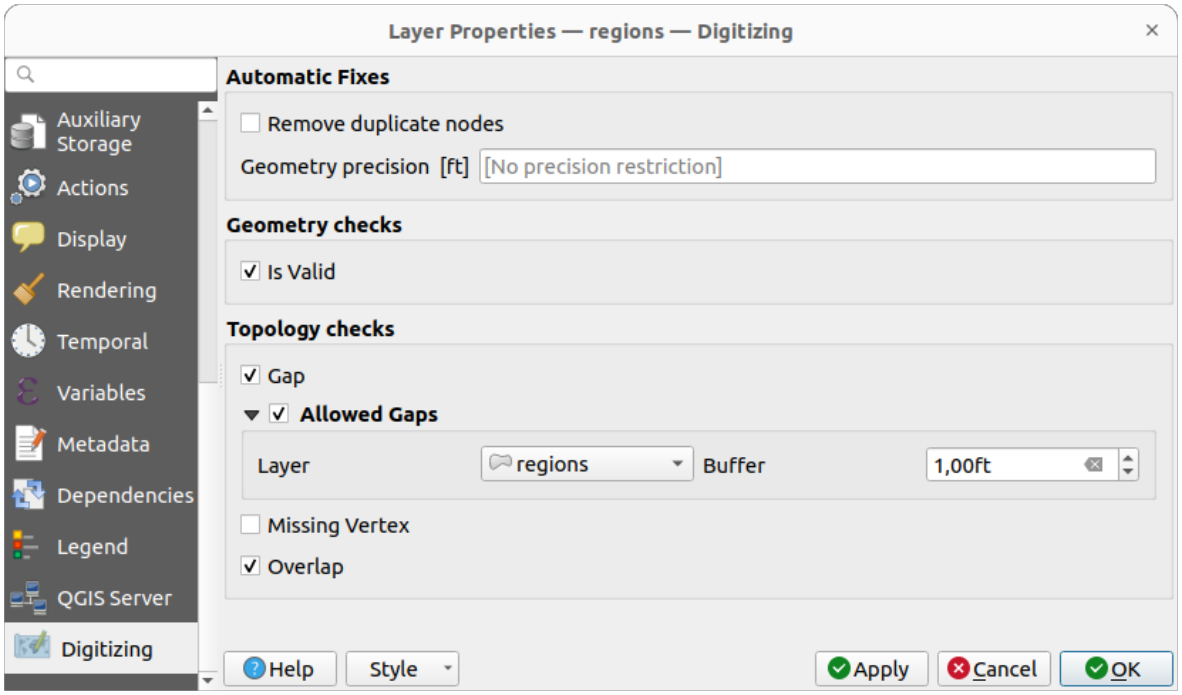


図 16.65: QGIS のベクタレイヤプロパティダイアログのデジタイズタブ

自動修正

自動修正 セクションのオプションは、追加または修正されたジオメトリの頂点に直接影響を与えます。 **重複ノードを削除** オプションがチェックされている場合、全く同じ座標の連続する 2 頂点は削除されます。**ジオメトリの精度** が設定されている場合、すべての頂点座標は設定されたジオメトリ精度の最も近い倍数に丸められます。丸めはレイヤの座標参照系で行われます。Z 値と M 値は丸められません。多くのマップツールでは、デジタイズ中にはキャンバスにグリッドが表示されます。

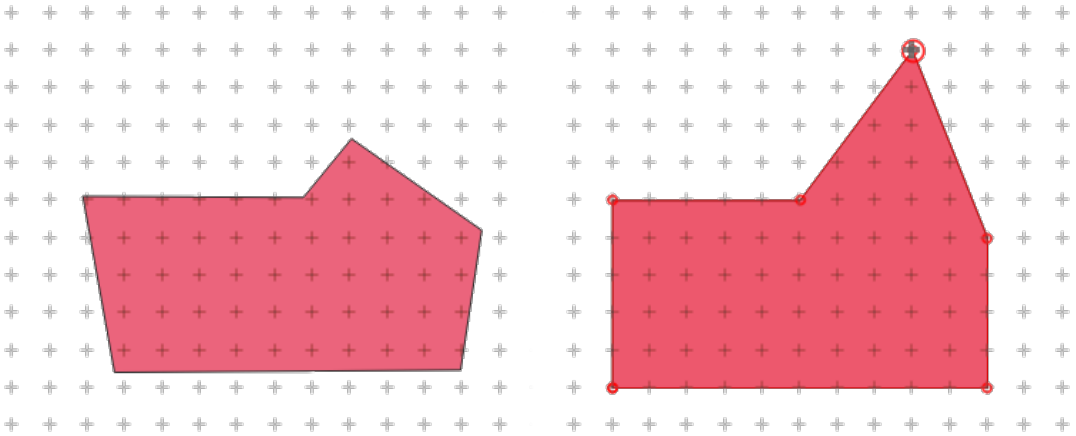


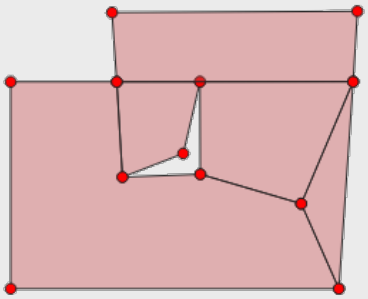
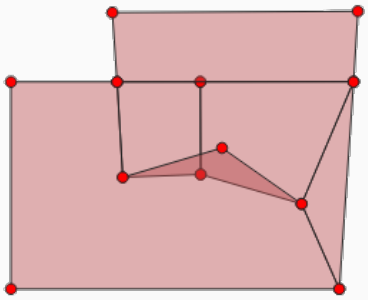
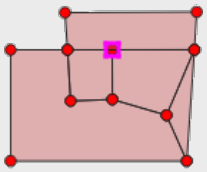
図 16.66: 上の頂点を移動させると、全ての頂点がグリッドにスナップする

ジオメトリチェック

ジオメトリチェック のセクションでは、ジオメトリ単位での追加的な検証を有効にできます。ジオメトリを変更すると直ちに、チェック失敗が **ジオメトリ検証** パネルでユーザーに報告されます。チェックに失敗している間は、レイヤを保存することができません。 **有効（*valid*）か** のチェックは、ジオメトリの自己交差のような、基本的な妥当性確認を実行します。

トポロジチェック

トポロジチェック のセクションでは、トポロジの追加的な検証を有効にできます。トポロジのチェックはユーザーがレイヤを保存した際に実行されます。チェックしたエラーは **ジオメトリ検証** パネルに報告されます。検証エラーが存在する限りは、レイヤを保存することができません。トポロジのチェックは、修正のあった地物のバウンディングボックス領域内で実行されます。同じ領域内に他の地物が存在することもあるため、現在の編集セッションで発生したエラーだけでなく、他の地物に関するトポロジエラーも報告されます。

トポロジチェックのオプション	説明図
☑ **隙間がないか** は、隣接するポリゴンとの間に隙間がないかをチェックします。	
☑ **重なり（*Overlap*）** は、隣接するポリゴンとの重なりがないかをチェックします。	
☑ **頂点欠損（点で分割されるべき直線）** のチェックは、隣接するポリゴンの共有された境界について、頂点が一方の境界には無く、他方の境界にはあるようなものをチェックします。	

隙間チェックの例外

他の部分はポリゴンで完全に覆われているものの、ポリゴンレイヤのある領域内には隙間を作っておくのが望ましい場合もあります。例えば、土地利用レイヤでは湖のために穴があり得るかもしれません。隙間チェックでは、無視される領域を定義することができます。このような領域内の隙間は許容されるものであるから、ここでは **許容される隙間（*Allowed Gaps*）** 領域と呼ぶことにします。

許容される隙間 の隙間チェックに関するオプションでは、 **レイヤ** を設定できます。

隙間チェックが実行される際に、 **許容される隙間のレイヤ** の 1 つ以上のポリゴンでカバーされている隙間はトポロジエラーとしては報告されません。

また、 **バッファ** を追加設定することもできます。このバッファは **許容される隙間のレイヤ** の各ポリゴンに適用されます。これにより、隙間の境目にあるアウトラインのわずかな変化に影響されにくいテストが可能となります。

許容される隙間 を有効にすると、検出された隙間エラーのための追加ボタン（ **許容される隙間を追加** ）が、デジタイズ中に隙間が報告されるジオメトリ検証ドックで利用できます。 **許容される隙間を追加** ボタンを押すと、検出された隙間のジオメトリを持つ新しいポリゴンが **許容される隙間レイヤ** に挿入されます。これにより、許容されている隙間に素早くフラグを立てることができます。

ジオメトリ検証パネル

ジオメトリ検証 パネルは、上記のデジタイズのチェックのいずれかによるエラーの検出が引き金となって表示されます。このダイアログにはエラーのリストとその説明が表示され、キーボードの矢印キーや専用の矢印ボタンを使用してリストをブラウズできます。

レイヤに編集内容を保存する前に、すべての問題に対処する必要があります。対処するためには：

1. エラーを一つ選択し、以下のボタンを利用できます：
   - **地物にズーム**
   - **問題箇所にズーム**
2. 通常どおり、 **デジタイジングツール** を選択して問題を修正します。

16.2 属性テーブルの操作

属性テーブルは選択されたレイヤの地物の情報を表示します。各行は（ジオメトリを持つ持たないに関わらず）1つの地物を表し、各列はその地物のある特定の情報を持っています。属性テーブルの地物は検索、選択、移動および編集が可能です。

16.2.1 序文: 空間情報のあるテーブル、空間情報のないテーブル

QGIS では、空間レイヤと非空間レイヤを読み込むことができます。これには現在、GDAL や区切りテキストでサポートされているテーブル、PostgreSQL、MS SQL Server、SpatiaLite、Oracle プロバイダが含まれます。読み込まれたすべてのレイヤは **レイヤ** パネルに一覧表示されます。レイヤが空間的に有効かどうかによって、マップ上でそのレイヤを操作できるかどうかが決まります。

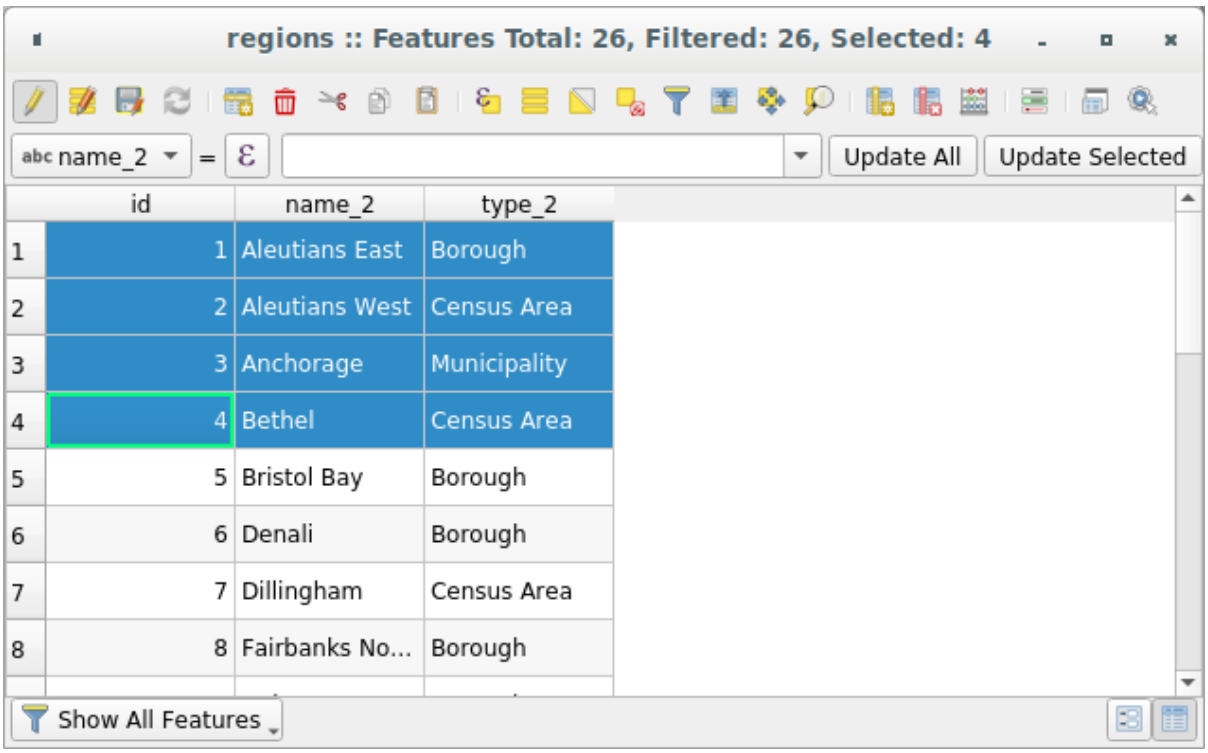
空間情報を持たないテーブルは、属性テーブルビューを使用して閲覧や編集ができます。さらに、これをフィールドの参照値に使用することも可能です。例えば、空間情報を持たないテーブルの列を使用して、編集モード中の特定のベクタレイヤに追加される属性値や許容される値の範囲を定義することができます。詳しくは、 **属性フォームプロパティ** セクションの編集ウィジェットを参考にしてください。

16.2.2 属性テーブルのインタフェースの紹介

ベクタレイヤの属性テーブルを開くには、 **レイヤパネル** 内でレイヤをクリックしてアクティブにします。次に、メインメニューの **レイヤ** メニューから、 **属性テーブルを開く** を選択します。レイヤを右クリックして、メニューから **属性テーブルを開く** を選択するか、属性ツールバーの **属性テーブルを開く** ボタンをクリックしても開くことができます。ショートカットキーがお好みなら、 F6 を押すと属性テーブルが開きます。 Shift+F6 は、選択した地物でフィルタリングされた属性テーブルを開き、 Ctrl+F6 は表示されている地物のみでフィルタリングされた属性テーブルを開きます。

これにより、レイヤの地物属性を表示するウィンドウが開きます（ *figure_attributes_table* ）。 **設定　オプション　データソース** メニューの設定により、属性テーブルはドックウィンドウとして開くか、または通常のウィンドウで開きます。レイヤ内の地物数の合計や、選択中の地物数、フィルタリングされた地物

数が属性テーブルのタイトルに表示されます。また、レイヤが空間的に制限されているかどうかも表示します。



	id	name_2	type_2
1	1	Aleutians East	Borough
2	2	Aleutians West	Census Area
3	3	Anchorage	Municipality
4	4	Bethel	Census Area
5	5	Bristol Bay	Borough
6	6	Denali	Borough
7	7	Dillingham	Census Area
8	8	Fairbanks No...	Borough

図 16.67: 地域レイヤの属性テーブル

属性テーブルウィンドウの上部にあるボタンには以下の機能があります。

表 16.1: 利用可能なツール

アイコン	ラベル	目的
	編集モード切替	編集機能を有効にする
	マルチエディットモード切替	多くの地物の複数のフィールドを更新する
	編集内容の保存	現在の修正を保存する
	テーブルを再読み込み	
	地物追加	ジオメトリを持っていない地物を新規追加します
	選択地物の削除	選択された地物をレイヤから削除します
	選択行を切り取ってクリップボードへ	
	選択している行をクリップボードへコピーする	
	クリップボードから地物を貼り付ける	コピーされた地物を新規地物として挿入します
	式による地物選択	
	すべて選択	レイヤ内のすべての地物を選択します
	選択部分を反転する	レイヤ内の現在の選択状態を反転します

表 16.1 – 前のページからの続き

アイコン	ラベル	目的
	レイヤ内の全地物を選択解除	現在のレイヤ内のすべての地物の選択を解除します
	フォームによる地物選択/フィルタ	
	選択を一番上に	選択した行をテーブルの先頭行に移動します
	選択した行の地物にパン	
	選択した行の地物にズームする	
	新規フィールド	データソースに新しいフィールドを追加します
	フィールド削除	データソースからフィールドを削除します
	列の整理	属性テーブルにフィールドを表示させたり、隠したりします
	フィールド計算機を開く	多数の地物のフィールドを連続的に更新します
	条件付き書式	表の書式設定を有効にします
	属性テーブルをドッキング	属性テーブルをウィンドウにドッキング/ドッキング解除し
	アクション	レイヤに関連したアクションをリストします

注釈: データの形式やお使いの QGIS のバージョンでビルドされた GDAL ライブラリによっては、利用できないツールがあります。

これらのボタンの下には、クイックフィールド計算バー（**編集モード** でのみ有効）があり、レイヤ内の地物のすべてまたは一部に計算結果を素早く適用できます。このバーは  フィールド計算機 と同様の **式** を使用します（**属性値の編集** 参照）。

属性テーブルは、Shift+マウスホイール スクロールに対応しており、縦スクロールから横スクロールに切り替わります。

テーブル表示 vs フォーム表示

QGIS は属性テーブル内のデータを簡単に操作するための 2 つの表示モードが用意されています。

-  テーブル表示 は、複数の地物の値を表形式で表示します。各行は 1 つの地物を、各列はフィールドを表しています。列のヘッダを右クリックすると **テーブルの列表示の設定** ができ、セルを右クリックすると、 **地物とのやり取り** ができます。
-  フォーム表示 は、最初のパネルに **地物の表示名** が表示され、クリックした表示名の属性のみを 2 つ目のパネルに表示します。最初のパネルの上部には、属性（ **カラムプレビュー** ）や **式** を使用して「表示名」を指定することができるプルダウンメニューがあります。このプルダウンメニューには、再利用のために最後に使用した 10 個の式も含まれています。フォーム表示はレイヤのフィールドの設定（ **属性フォームプロパティ** 参照）を使用します。

  最初のパネルの下部にある矢印ボタンを使用して、地物 ID でブラウジングができます。2 番目のパネルに表示される地物の属性は、矢印ボタンを押すたびに順に更新されます。また、下部にあるボタ

ンを押すことで、マップキャンバスでアクティブな地物を識別したり、アクティブな地物へ画面を移動することもできます。

- マップキャンバスに地物が表示されている場合に、 現在の地物をハイライト表示
- 現在の地物に合わせて縮小表示
- 現在の地物に合わせて拡大

あるモードから他方のモードへの切り替えは、ダイアログの右下にある対応するアイコンをクリックすることによって行えます。

設定　オプション　データソース メニューで、属性テーブルのオープン時の **デフォルトビュー** モードを指定することもできます。「直前の表示形式」、「テーブル表示」または「フォーム表示」にすることができます。

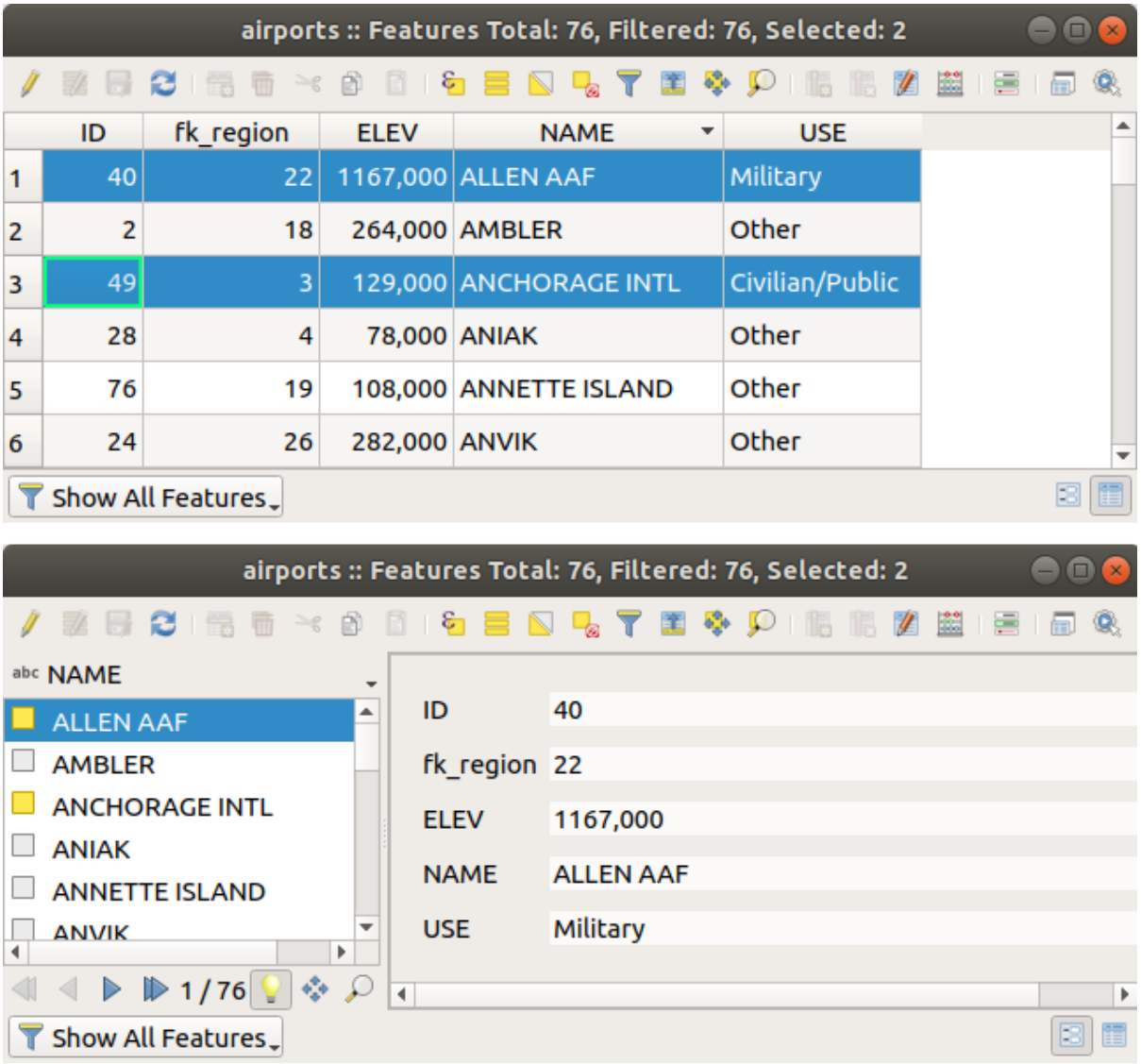


図 16.68: 属性テーブルのテーブル表示（上）とフォーム表示（下）

列の設定

テーブル表示の場合に列のヘッダを右クリックすると、以下の項目をコントロールするツールにアクセスできます：

- **列の幅**
- **列の可視性と順序**
- **データのソート順序**

列の幅の修正

列の幅は、列見出しを右クリックして、次のいずれかを選択することで設定できます：

- **幅の設定**... 希望する幅の値を入力します。デフォルトでは、現在の値がウィジェットに表示されます
- **全カラム幅を設定**... は、すべてのカラムの幅を同じ値で設定します
- **自動サイズ** は、列に合わせて最適なサイズに変更します。
- **全カラム幅を自動設定**

カラムの幅は、カラムのヘッダの右側の境界をドラッグすることでも変更できます。新しいカラム幅はレイヤに保持され、次に属性テーブルを開いたときにこの幅が復元されます。

列の非表示・整理とアクションの有効化

列のヘッダを右クリックすると、属性テーブルから **列を非表示** とするかどうかを選択できます（「テーブル表示」の場合）。より詳細なコントロールについては、ダイアログのツールバーの  列の整理... ボタンを押すか、列のヘッダの右クリックメニューで **列の整理**... を選択します。このダイアログでは、以下のことができます：

- 表示したい列にチェック、または非表示したい列のチェックを外す：非表示にした列は、自ら復元するまで、属性テーブルダイアログのすべてのインスタンスから消えます。
- アイテムをドラッグ＆ドロップして、属性テーブル内の列の順序が変更できます。この変更はテーブルの表示のためのものであり、レイヤのデータソースのフィールドの順序は変更されないことに注意してください。
- 新しい仮想の **アクション** 列を追加して、各行にドロップダウンボックスを表示したり、利用可能なアクションのボタンリストを表示します。アクションに関する更なる情報については、 **アクションプロパティ** を参照してください。

列による並べ替え

テーブルは列のヘッダをクリックすることで任意の列でソートできます。小さな矢印はソート順を示しています（下向きは先頭行から値を降順で、上向きは先頭行から値を昇順で並べることを意味します）。列のヘッダのコンテキストメニューの **並べ替え**... オプションを使用して式を書くことでも行を並べ替えることができます。例えば、複数の列を使用して行を並べ替えるには `concat(col0, col1)` と書きます。

フォーム表示では、地物識別子は **プレビュー式で並べ替え** オプションを使用してソートできます。

Tip: 異なる型の列に基づいた並べ替え

文字列型の列と数値型の列に基づいて属性テーブルを並べ替えようとすると、予期しない結果となることがあります。これは、式 `concat("USE", "ID")` が文字列値を返す（つまり `'Borough105' < 'Borough6'` である）ために起こります。この問題を回避するには、例えば、 `concat("USE", lpad("ID", 3, 0))` とすると、 `'Borough105' > 'Borough006'` を返します。

条件式を使ったテーブルセルの書式設定

条件付き書式設定を使用して、属性テーブル内で特に注目したい地物を強調表示させることができます。カスタム条件式は、地物に関する以下の要素に基づきます：

- ジオメトリ（例：マルチパート地物、面積が小さい地物、定義された地図範囲内にある地物などを識別する）
- フィールド値（例：しきい値との比較や、空のセルを区別するなど）

テーブル表示の属性ウィンドウの右上にある をクリックして、条件付き書式パネルを有効にできます（フォーム表示では使用できません）。

この新しいパネルでは、 **フィールド** または **行全体** の書式レンダリングに新ルールを追加することができます。新ルールを追加すると、以下のものを定義するためのフォームが開きます：

- ルールの名前
- 何らかの **式ビルダー** 関数を使用した条件式
- 書式設定：プリセット書式のリストから選択するか、以下のプロパティに基づいて作成できます。
  - 背景色とテキスト色
  - アイコンを使用するかどうか
  - 太字、イタリック、下線、取り消し線
  - フォント

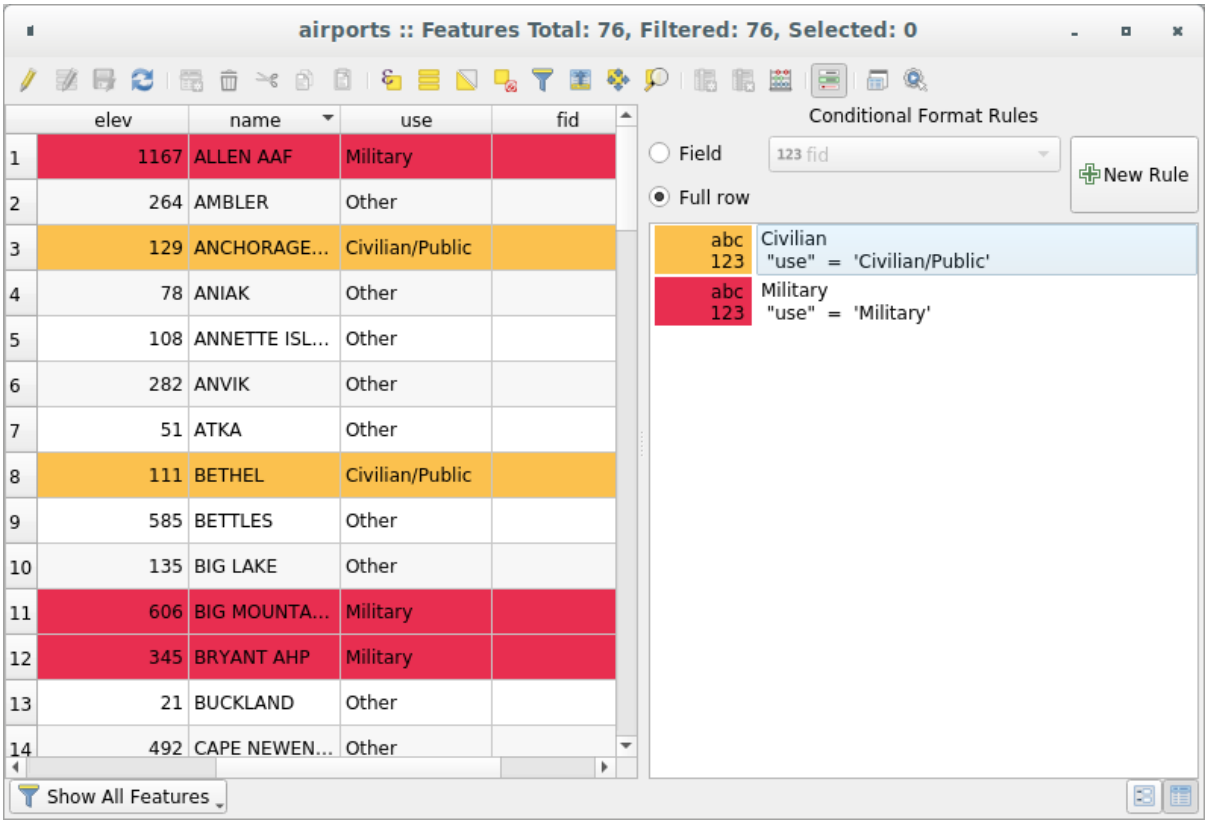


図 16.69: 属性テーブルの条件付き書式

16.2.3 属性テーブルの地物とのやりとり

地物の選択

テーブル表示では、属性テーブルの各行はレイヤ内の個別地物の属性を表します。ある行を選択するとその地物が選択され、同様に、（ジオメトリのあるレイヤの場合は）マップキャンバスで地物を選択すると属性テーブル内の行が選択されます。マップキャンバス（または属性テーブル）で選択された地物の組み合わせが変更された場合、それに応じて属性テーブル（またはマップキャンバス）の選択が更新されます。

行の左端にある行番号をクリックすることで、行を選択することができます。 `Ctrl` キーを押しながらクリックすると、 **複数の行** をマークすることができます。 `Shift` キーを押しながら行の左側にある複数の行ヘッダをクリックすると、 **連続選択** ができます。現在のカーソル位置とクリックした行の間のすべての行が選択されます。属性テーブルのセルをクリックしてカーソル位置を移動しても、行の選択は変わりません。また、メインキャンバス内で選択を変更しても、属性テーブルのカーソル位置は移動しません。

属性テーブルのフォーム表示では、地物はデフォルトで左側パネル内において表示フィールド（ **表示名プロパティ** 参照）の値によって識別されています。この識別子はパネルの上部にあるドロップダウンリストを使用して、既存のフィールドを選択するか、カスタム式を使用して置き換えることができます。また、ドロップダウンメニューから地物フォームのリストをソートすることもできます。

左側のパネルで値をクリックすると、右側のパネルにその地物の属性が表示されます。地物を選択するには、識別子の左にある四角いシンボルの中をクリックします。デフォルトでは、シンボルが黄色に変わり

ます。テーブル表示と同様、これまでに紹介したキーボードの組み合わせで複数の地物を選択することができます。

マウスで地物を選択するだけでなく、属性テーブルのツールバーにある以下のようなツールを使用して、地物の属性に基づいた自動選択を行うことができます（詳細や使用例については、**自動選択** および次のセクションを参照）：

-  **式による地物選択...**
-  **値による地物選択...**
-  **レイヤ内の全地物を選択解除**
-  **すべて選択**
-  **選択部分を反転する**

フォームによるフィルタと地物選択 を使って地物を選択することもできます。

地物のフィルタリング

属性テーブルで地物を選択したら、テーブルにこれらのレコードのみを表示したい場合があります。これは、属性テーブルダイアログの左下にあるドロップダウンリストから **選択した地物を表示** アイテムを使用すると簡単にできます。このリストには、次のようなフィルタがあります：

-  **全地物を表示**
-  **選択した地物を表示** - これは、 **レイヤ** メニューや **属性ツールバー** から **属性テーブルを開く（選択地物）** を使用する場合や、キーボードで `Shift+F6` を押した場合と同じです
-  **地図上に表示されている地物を表示** - これは、 **レイヤ** メニューや the **属性ツールバー** から **属性テーブルを開く（可視地物）** を使用する場合や、キーボードで `Ctrl+F6` を押した場合と同じです
-  **編集された地物と新規地物を表示** - これは、 **レイヤ** メニューや **属性ツールバー** から **属性テーブルを開く（編集済み地物）** を使用する場合と同じです
- **属性フィルタ** - フィールドの値に基づいたフィルタリングができます：リストからカラムを選択し、値を入力するか選択して、`Enter` キーを押すとフィルタリングが実行されます。すると、`num_field = value` の式や `string_field ilike '%value%'` の式にマッチする地物のみが属性テーブルに表示されます。文字列に関しては  *Case sensitive* にチェックを入れることで、大文字小文字を区別するより厳しいマッチングにできます。
- **詳細フィルタ（式）** - 式ビルダーダイアログを開きます。ここでは、テーブルの行にマッチさせるための **複雑な式** を作成することができます。例えば、複数のフィールドを使用してテーブルをフィルタできます。OK ボタンを押すと、そのフィルタ式がフォームの下部に表示されます。
- **保存済みフィルタ式**　属性テーブルのフィルタリングによく使用する **保存済み式** へのショートカットです。

これらは **フォームによる地物フィルタ** でも利用可能です。

注釈: 属性テーブルからレコードをフィルタリングしても、レイヤから地物がフィルタリングされるわけではありません。地物はテーブルから一時的に非表示になるだけで、マップキャンバスからはアクセスでき、フィルタを取り除いてもアクセスすることができます。フィルタでレイヤから地物を実際に隠すには、**クエリビルダ** を使用してください。

Tip: 地図上に表示されている地物を表示 **によるデータソースフィルタリングの更新**

パフォーマンス上の理由で、属性テーブルに表示される地物がテーブルを開いたときのキャンバス範囲に空間的に制限されている場合（設定方法は **データソースオプション** を参照）、新しいキャンバス範囲で **地図に表示されている地物を表示** を選択すると、空間的な制限が更新されます。

フィルタ式の保存

属性テーブルのフィルタリングに使用した式は、以降の呼び出しのために保存することができます。 **属性フィルタ** や **詳細フィルタ（式）** のエントリを使用すると、使用されている式が属性テーブルダイアログの下部にあるテキストウィジェットに表示されます。テキストボックスの横にある 名前付きで式を保存 を押すと、プロジェクトに式が保存されます。ボタンの横にあるドロップダウンメニューを使用して、自分で式の名前を付けて保存できます（ **式を新たに保存...** ）。保存された式が表示されたら、 ボタンがトリガーされ、そのドロップダウンメニューから **式を編集** （式の名前も編集可）や、 **保存済み式の削除** が行えます。

保存済みフィルタ式はプロジェクト内に保存され、属性テーブルの **保存済みフィルタ式** メニューをから利用できます。これは、アクティブなユーザープロファイルの全プロジェクトで共有される **ユーザー保存式** とは異なります。

フォームによるフィルタと地物選択

フォームによる地物選択/フィルタ をクリックするか、または `Ctrl+F` を押すと、属性テーブルダイアログをフォーム表示に切り替え、各ウィジェットが検索変数に置き換わります。

ここで説明するこのツールの機能は、 **値による地物選択** で説明されているものと同様です。そちらには、すべての演算子と選択モードについての説明があります。

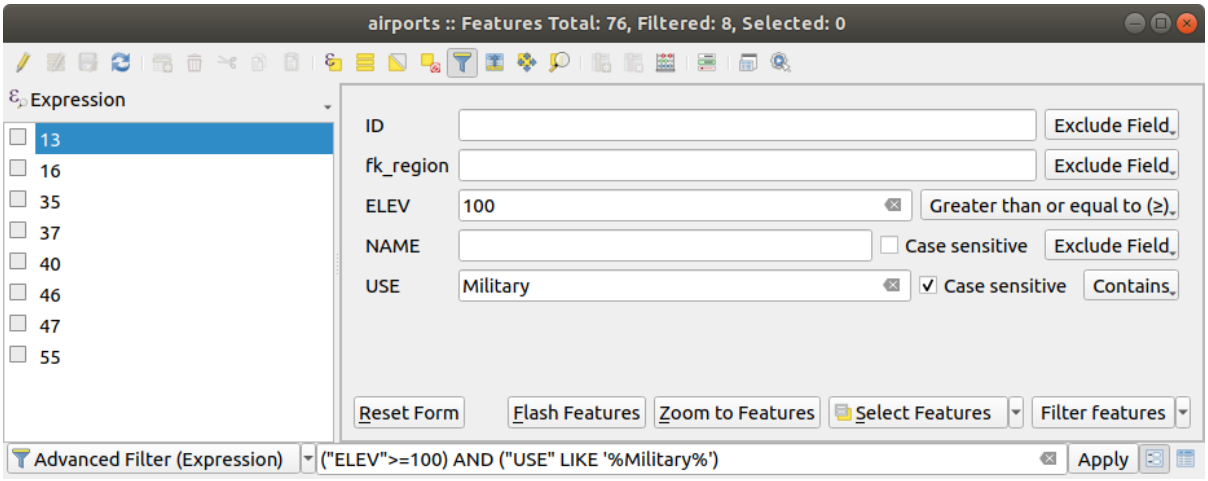


図 16.70: フィルタフォームでフィルタされた属性テーブル

属性テーブルから地物を選択/フィルタする際には、フィルタを定義したり、絞り込んだりできる **地物をフィルタする** ボタンがあります。このボタンを押すと **詳細フィルタ（式）** オプションがトリガーされ、対応したフィルタ式がフォームの下部にある編集可能なテキストウィジェットに表示されます。

すでにフィルタされた地物がある場合は、 **地物をフィルタする** ボタンの隣にあるドロップダウンリストを使用してフィルタを絞り込むことができます。オプションは次のとおりです。

- **フィルタの絞り込み** *("AND")*
- **フィルタを拡げる** *("OR")*

フィルタを削除するには、左下のプルダウンメニューから **全地物を表示** オプションを選択するか、または式をクリアして **適用** ボタンをクリックするか `Enter` キーを押します。

16.2.4 地物に関するアクション

ユーザーはコンテキストメニューを使用していくつかの地物操作が可能です：

- 地物を **すべて選択** （ `Ctrl+A` ）
- **セルの内容をコピーする** で、セルの内容をクリップボードにコピーする
- 地物を選択することなく **地物にズーム**
- 地物を選択することなく **地物にパン**
- **地物をフラッシュ** で、マップキャンバス内で強調表示
- **フォームを開く** ：そのクリックされた地物にフォーカスした状態で、属性テーブルをフォーム表示に切り替えます

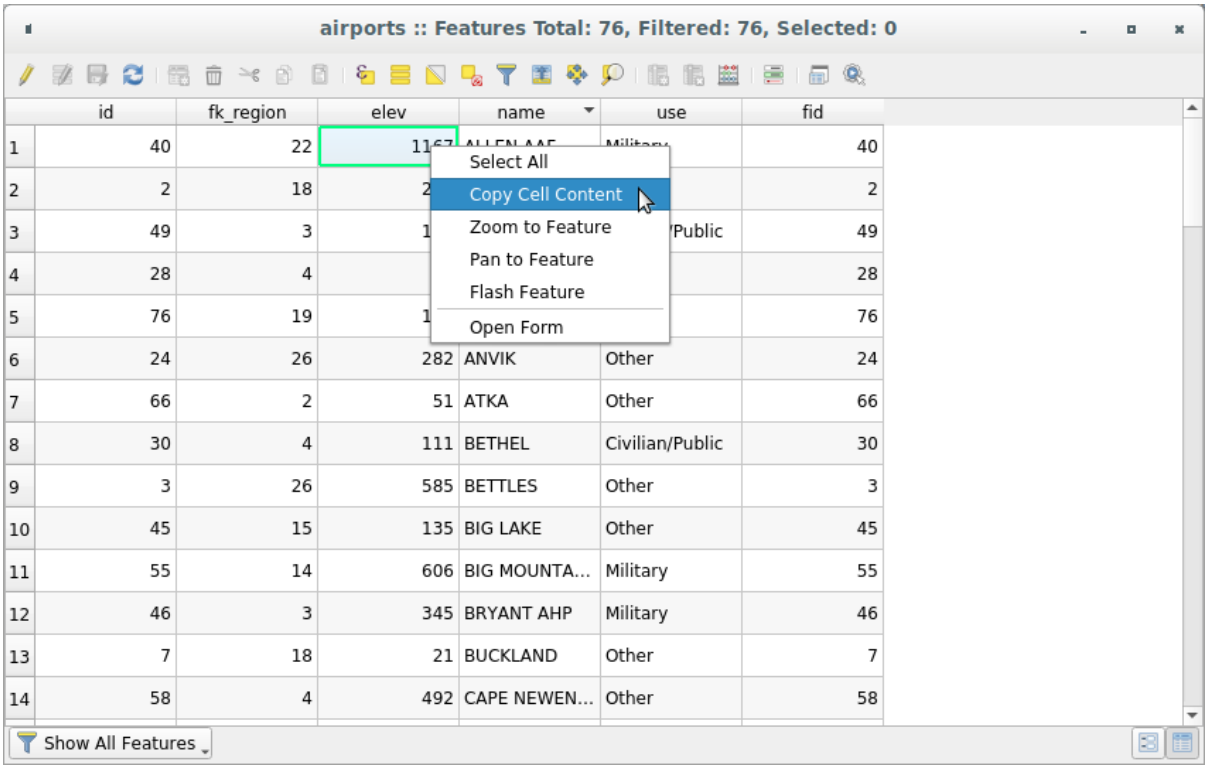


図 16.71: 「セルの内容をコピーする」ボタン

属性データを外部プログラム（Excel、LibreOffice、QGIS、カスタム Web アプリケーションなど）で使用したい場合には、1 つまたは複数の行を選択して 選択している行をクリップボードへコピーする ボタンを押すか、`Ctrl+C` を押します。

設定　オプション　データソース メニューでは、 **地物のコピー** ドロップダウンリストで貼り付け形式を指定できます：

- プレーンテキスト（ジオメトリなし）
- プレーンテキスト（WKT ジオメトリ）
- GeoJSON

このコンテキストメニューでアクションのリストを表示することもできます。これは **レイヤプロパティ　アクション** タブから有効にできます。アクションの詳細については **アクションプロパティ** を参照してください。

選択地物を新規レイヤとして保存する

選択された地物は、OGR がサポートする任意のベクトル形式として保存でき、別の座標参照系 (CRS) に変換することもできます。**レイヤ** パネルでレイヤのコンテキストメニューから **エクスポート　新規ファイルに選択地物を保存...** をクリックし、出力データセットの名前、形式、CRS を定義します（セクション **既存のレイヤから新しいレイヤを作成する** を参照）。 **選択地物のみ保存** がチェックされていることが分かります。ダイアログ内で GDAL の作成オプションを指定することも可能です。

16.2.5 属性値の編集

属性値の編集は以下の方法で行うことができます。

- 新しい値をセルに直接入力します。属性テーブルがテーブル表示でもフォーム表示でも可能です。従って、変更はセルごと、地物ごとに行われます。
- **フィールド計算機** を使用して、あるフィールド列全体に対して連続して更新を行います。フィールドは既存のものでも、新しく作られたものでもよいですが、更新は複数の地物に対して行われます。これは仮想フィールドを作るためにも使用することができます。
- クイックフィールド **計算バー** を使用します。上と同じですが既存のフィールドに対してのみ使用できます。
- **マルチエディット** モードを使用します。複数の地物の複数のフィールドを連続して更新します。

フィールド計算機を使用する

属性テーブルの  フィールド計算機 ボタンを使うと、既存の属性値もしくは定義された関数に基づいて計算を行うことができます。例えば、ジオメトリ地物の長さや面積の計算ができます。計算結果は既存のフィールドの更新や、新しいフィールド（これは **仮想** フィールドも可）への書き込みに使うことができます。

フィールド計算機 は編集をサポートするすべてのレイヤで利用可能です。フィールド計算機のアイコンをクリックすると、ダイアログを開きます (図 16.72 参照)。レイヤが編集モードでない場合には警告が表示され、フィールド計算機を使用すると、計算が行われる前にレイヤが編集モードに変わります。

式ビルダー ダイアログをベースにしたフィールド計算機ダイアログは、式を定義し、それを既存のもしくは新規作成フィールドに適用するための完璧なインターフェイスを提供します。フィールド計算機ダイアログを使うには、以下について選択する必要があります：

1. レイヤの全体に計算を適用したいのか、それとも選択した地物のみに適用したいのか
2. 計算によって新しいフィールドを作りたいのか、それとも既存のフィールドを更新したいのか

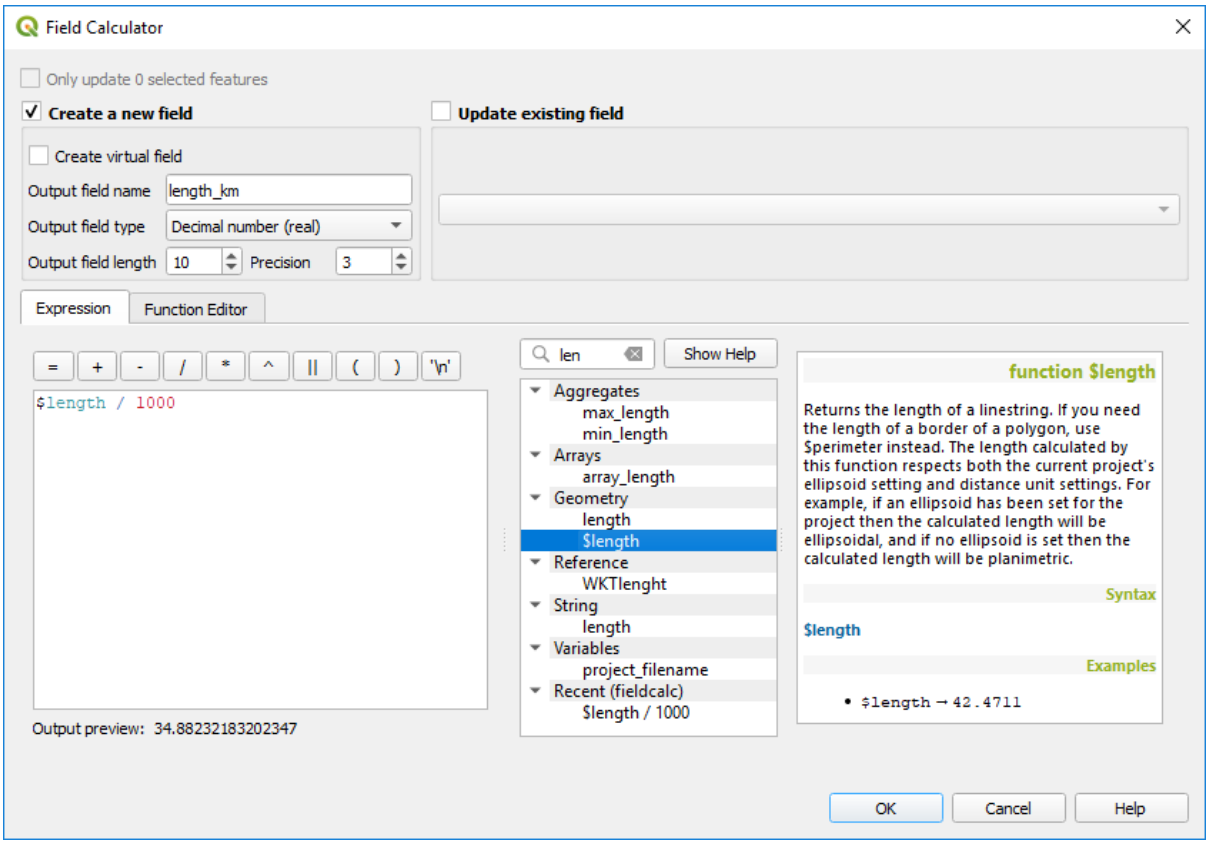


図 16.72: フィールド計算機

新しいフィールドを作ることを選んだ場合には、フィールドの名前やフィールド型（整数値、小数点付き数値、日付、テキストなど）を入力する必要があり、必要ならばフィールド長や精度も入力します。例えば、フィールド長が 10 で精度を 3 とすると、これは小数点より前に 7 桁あり、小数部が 3 桁の数値ということになります。

式 タブを使用している場合のフィールド計算機の動作を簡単な例で説明します。QGIS サンプルデータセットから、 `railroads` レイヤの長さを km 単位で計算したいとします：

1. QGIS でシェープファイル `railroads.shp` をロードし、 属性テーブルを開く を押します。
2. 編集モード切替 をクリックし、 フィールド計算機 ダイアログを開きます。
3. **新しいフィールドを作成** チェックボックスを選択し、新しいフィールドに計算を保存するようにします。
4. **出力する属性（フィールド）の名前** を `length_km` に設定します。
5. **フィールド型** として 小数点付き数値（`real`） を選択します。
6. **フィールド長** を `10` に、 **精度** を `3` に設定します。
7. **ジオメトリ** グループにある `$length` をダブルクリックして、ジオメトリの長さをフィールド計算機の式ボックスに追加します（式ボックスの下に最大 60 文字までの出力のプレビューが表示され、式が組み立てられるとリアルタイムに更新されます）。
8. フィールド計算機の式ボックスで `/ 1000` を入力して式を完成させ、 *OK* をクリックします。

9. これで、属性テーブルに新しい *length_km* フィールドが出来ました。

仮想フィールドの作成

仮想フィールドとは、オンザフライで計算される式に基づいたフィールドのことです。式の基礎となるパラメータが変更されると値が自動的に更新されます。式の設定は一度だけでよく、基礎となる値が変更されるたびにフィールドの再計算を行う必要はありません。例えば、地物をデジタイジングするたびに面積を評価する場合や、変更の可能性がある（例： `now()` 関数を使用する）日付間の期間を自動的に計算する場合などで、仮想フィールドを使用するのが良いでしょう。

注釈: 仮想フィールドの使用

- 仮想フィールドはレイヤの属性として永続的ではありません。つまり、仮想フィールドを作成したプロジェクトファイル内にのみ保存され、そこでのみ利用可能です。
- フィールドは作成時にのみ、仮想フィールドに設定することができます。仮想フィールドは、レイヤプロパティダイアログのフィールドタブに紫色の背景で表示され、通常の物理フィールドや結合されたフィールドとは区別できるようになっています。仮想フィールドの式は、「コメント」列の式ボタンを押すことで、後から編集することができます。ボタンを押すと式エディタウィンドウが開き、仮想フィールドの式の修正ができます。

クイックフィールド計算バーを使用する

フィールド計算機がいつでも利用可能なのに対して、属性テーブルの上部にあるクイックフィールド計算バーは、レイヤが編集モードにある場合にのみ表示されます。式エンジンのおかげで、クイックフィールド計算バーを使えば既存のフィールドの編集がより素早くできるようになります。

1. ドロップダウンリストから更新したいフィールドを選択します。
2. テキストボックスに直接書くか、  式ボタンを使用して値や式を入力します。
3. 必要に応じて、 **すべて更新** 、 **選択を更新** 、または **フィルタされたものを更新** ボタンをクリックします。

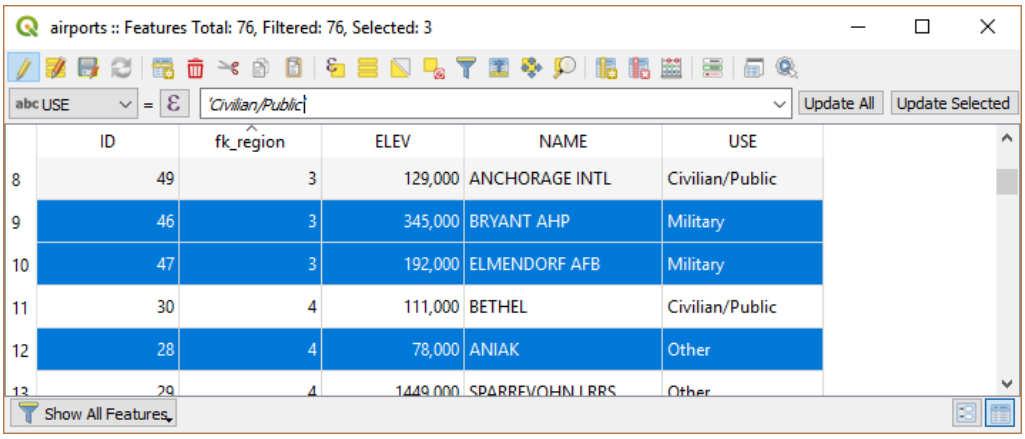


図 16.73: クイックフィールド計算バー

複数のフィールドの編集

これまでのツールとは異なり、マルチエディットモードでは、さまざまな地物の複数の属性を同時に編集できます。レイヤを編集モード切り替えて、以下の操作によりマルチエディット機能にアクセスできます。

- 属性テーブルダイアログ内のツールバーの マルチエディットモード切替 ボタンを押す
- または、 **編集** **選択地物の属性変更** メニューを選択する

注釈: 属性テーブルのツールとは異なり、 **編集** **選択地物の属性変更** オプションを選択すると現れる、変更する属性値を入力するためのダイアログはモーダルなダイアログです。したがって、実行前に地物の選択が必要です。

複数のフィールドを一度に編集するには：

1. 編集したい地物を選択します。
2. 属性テーブルツールバーから をクリックします。これによりダイアログがフォーム表示に切り替わります。地物の選択はこの段階でも行うことができます。
3. 属性テーブルの右側には、選択した地物のフィールド（と値）が表示されています。新しいウィジェットが各フィールドの横に表示され、現在のマルチエディット状態を表示します。
   - 選択した地物はフィールドに異なる値を有しています。値は空で表示され、各地物は元の値を持っています。ウィジェットのドロップダウンリストから、フィールドの値をリセットすることができます。
   - 選択した地物は全てフィールドに同じ値を持っており、フォームに表示される値を保持します。
   - フィールドの値が編集され、入力した値を選択した地物すべてに適用しようとする状態です。ダイアログの上部には、変更を適用するか、リセットするかを案内するメッセージが表示されます

これらのウィジェットのいずれかをクリックすると、フィールドの現在の値を設定するか、元の値にリセットできます。つまり、フィールド単位で変更をロールバックできます。

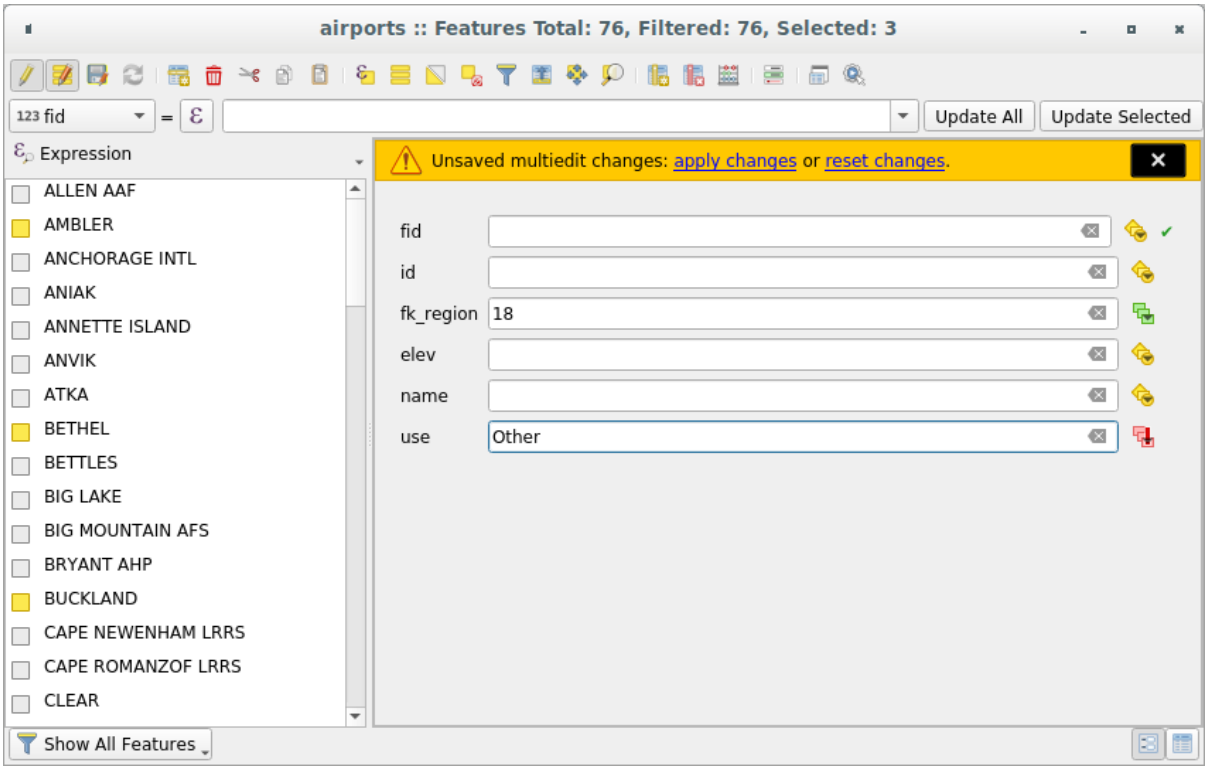


図 16.74: 複数地物のフィールドの編集

4. 変更したいフィールドを編集します。
5. 上部のメッセージ内の **変更の適用** をクリックするか、または左側パネルの別の地物をクリックします。

選択した地物すべて に変更が適用されます。地物が選択されていない場合、テーブル全体が変更内容で更新されます。変更は単一の編集コマンドとして行われます。したがって、元に戻す を押すと、選択したすべての地物の属性変更を一度にロールバックします。

注釈: 複数編集モードは「自動生成」または「ドラッグ&ドロップ」のフォームでのみ利用可能です（**データに合わせてフォームをカスタマイズする** 参照）。カスタム UI フォームではサポートされていません。

16.2.6 1 対多または多対多のリレーションの作成

「リレーション」は、データベースでよく使用される技術です。その概念は、異なるレイヤ（テーブル）の地物（行）がお互いに属することができるというものです。

1 対 N リレーションの導入

例として、アラスカのすべての地域（ポリゴン）を有するレイヤがあり、名前、地域タイプ、（主キーとして機能する）ユニーク ID といった属性を持っているとします。

それから、地域にある空港の情報を持った別のポイントレイヤまたはテーブルがあり、これらの情報も把握したいとします。ほとんどの地域には複数の空港があるため、これらを地域レイヤに追加する場合は、外部キーを使って 1 対多のリレーションを作成する必要があります。

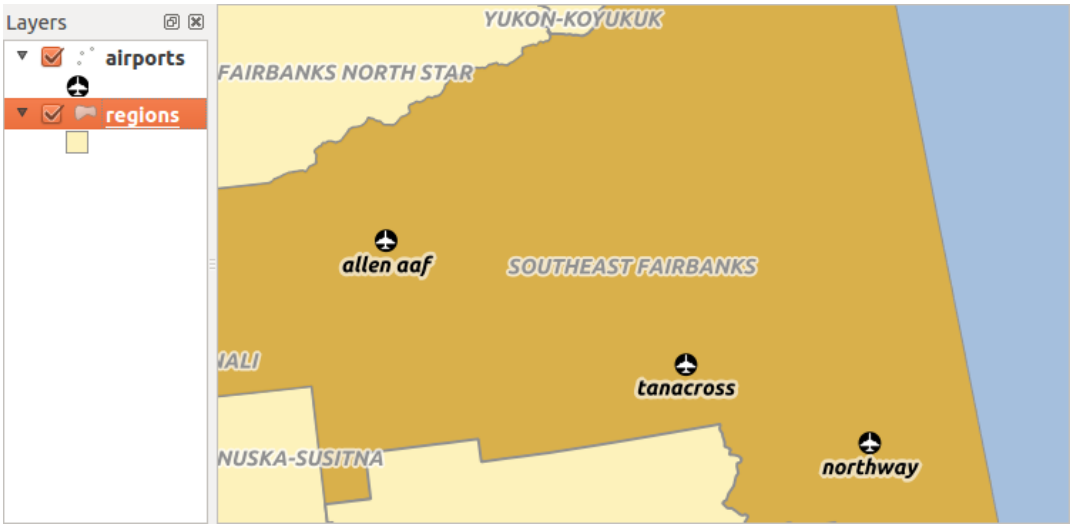


図 16.75: 空港とアラスカの地域

1 対 N リレーションのレイヤ

QGIS では、テーブルとベクタレイヤの間に違いはありません。基本的には、ベクタレイヤとはジオメトリを持っているテーブルのことです。このため、テーブルはベクタレイヤとして追加することができます。1 対 N リレーションのデモンストレーションとして、 `regions` シェープファイルと、地域レイヤへの外部キーフィールド（ `fk_region` ）を持つ `airports` シェープファイルを読み込んでみましょう。これは、各空港はそれぞれただ 1 つの地域に属し、各地域は空港をいくつでも持つことができることを意味します（典型的な 1 対多関係です）。

1 対 N リレーションの外部キー

空港の属性テーブルに既にある属性に加えて、外部キーとして動作する別のフィールド `fk_region` が必要です（データベースを持っている場合には、このフィールドに制約を定義することになるでしょう）。

このフィールド fk_region には、必ず地域の ID が入ります。これは、空港が属する地域へのポインタのように見ることができます。また、編集用のカスタム編集フォームを設計でき、QGIS はその設定を処理します。これはさまざまなプロバイダに対応しており（シェープファイルや CSV ファイルでも使うことができます）、しなければならないのは、QGIS にテーブル間の関係を伝えることだけです。

1 対 N リレーションの定義

まず最初に行うのは、レイヤ間の関係を QGIS に知らせることです。これは、**プロジェクト** ▸ **プロパティ**... で行います。 **リレーション** タブを開き、 **リレーションを追加** をクリックします。

- **名前** はタイトルとして使用されます。これはわかりやすい文字列で、リレーションが何のために用いられるのかを説明するものであるべきです。ここでは、単に **airport_relation** と名前を付けましょう。
- **被参照レイヤ（親）** 親レイヤとも呼ばれ、プライマリキーを持っており、参照されるレイヤです。この例では、 `regions` レイヤです。参照レイヤの主キーを定義する必要があり、ここでは `ID` を指定します。
- **参照元レイヤ（子）** 子レイヤとも呼ばれ、外部キーフィールドを持ったレイヤです。この例では、`airports` レイヤが該当します。参照元レイヤには他のレイヤを指す参照フィールドを指定する必要があり、ここでは `fk_region` とします。

注釈: 場合によっては、レイヤ内の地物を一意に識別するためには複数のフィールドが必要となることがあります。このようなレイヤとのリレーションを作成するためには、**複合キー** 、すなわち、一致するフィールドの複数ペアが必要です。 複合外部キーとしてペアのフィールドを追加 ボタンを使用して、必要な数のペアを追加できます。

- **Id** は内部的な目的で使用され、ユニークでなければなりません。これは **カスタムフォーム** を作成するために必要となる場合があります。空のままにしておくと自動で Id を作成しますが、扱いやすいように自分で Id を指定することもできます。
- **リレーションの強度** は、親子レイヤ間のリレーションの強さを設定します。デフォルトの *Association* タイプは、親レイヤが子レイヤに **単に** リンクされていることを意味します。 一方、 **コンポジション** タイプでは、親レイヤを複製するときに子レイヤも複製され、親レイヤの地物を削除すると子レイヤの地物も削除されます。これはすべてのレベルにカスケードしていきます（つまり、子レイヤのそのまた子レイヤ... も一緒に削除されます）。

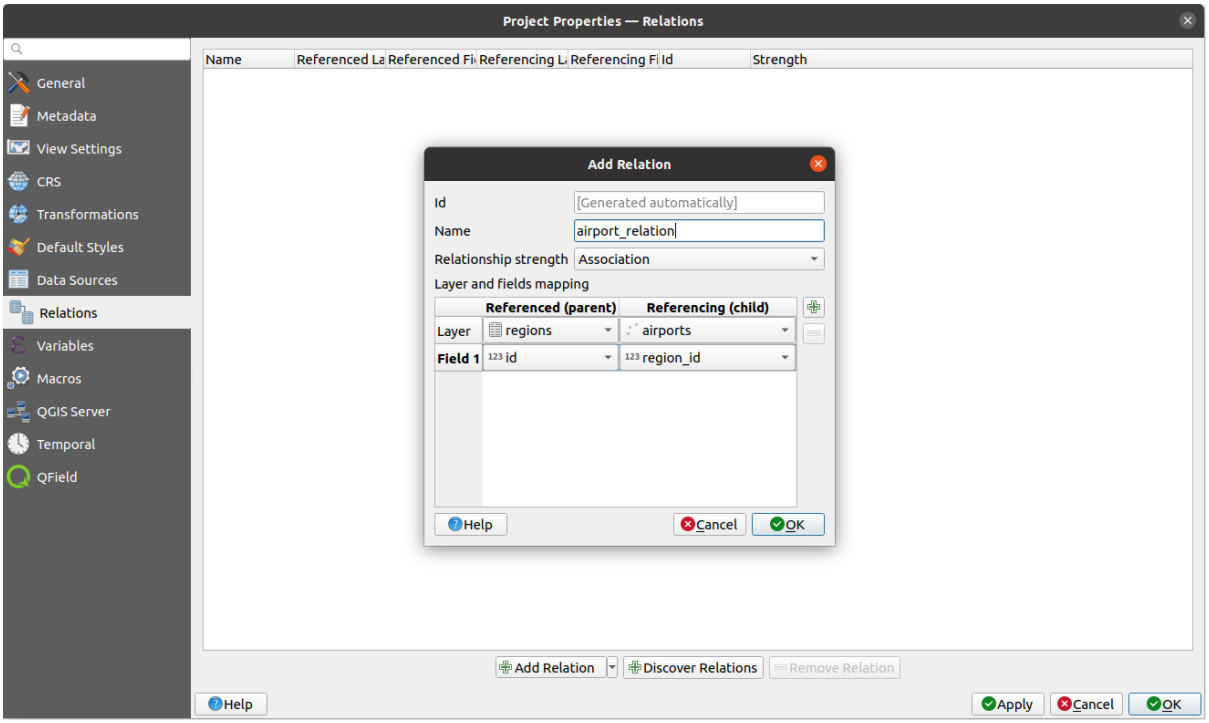


図 16.76: regions レイヤと airports レイヤ間のリレーションの追加

リレーション タブでは、 **リレーションを検索** ボタンを押して、読み込んだレイヤの利用可能なリレーションをプロバイダから取得することもできます。この機能は PostgreSQL や SpatiaLite といったデータプロバイダに保存されたレイヤで使用可能です。

1 対 N リレーションのフォーム

これで、QGIS はリレーションを把握できました。これは QGIS が生成するフォームを改善するために利用されます。デフォルトのフォームのメソッド（自動生成）を変更していないので、改善はフォームに新しいウィジェットが追加されるのみです。それでは、凡例の regions レイヤを選択して、識別ツールを使ってみましょう。設定によってはフォームが直接開くこともありますが、識別ダイアログのアクションから選択して地物フォームを開く必要があるかもしれません。

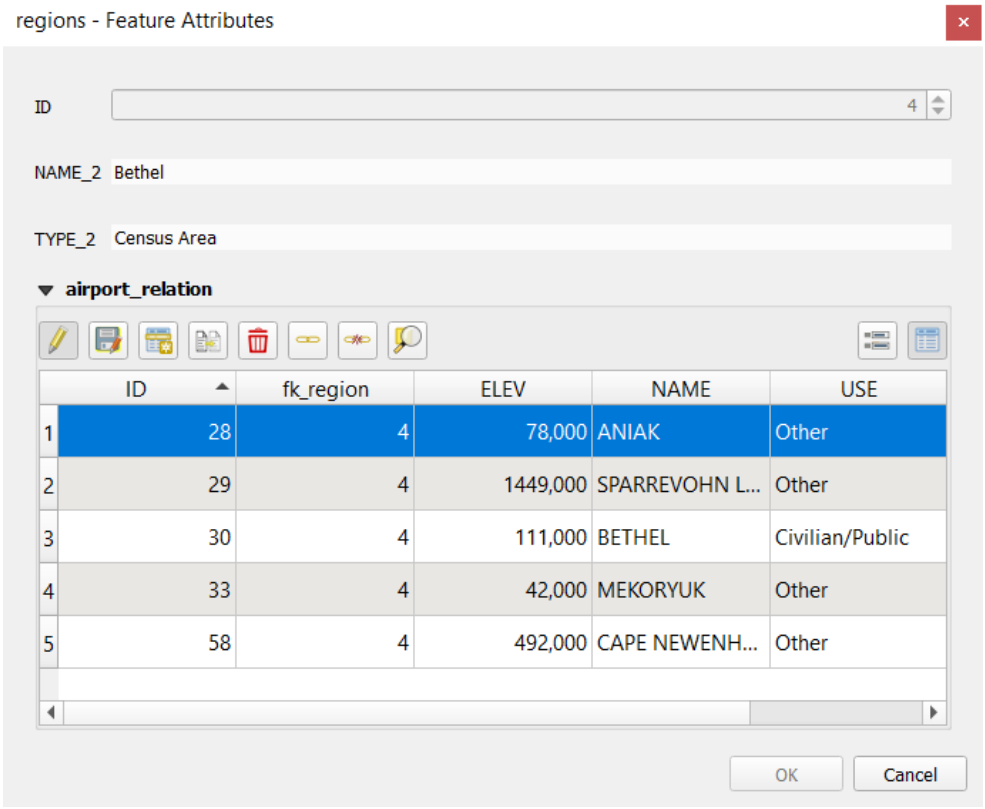


図 16.77: airports へのリレーションを持つ regions の識別ダイアログ

見ればわかるとおり、この特定の地域に割り当てられた空港が全てテーブル表示されています。また、いくつかのボタンも用意されています。早速見てみましょう。

- ボタンは編集モードを切り替えるためのものです。regions レイヤの地物の地物フォームを開いていますが、このボタンは airports レイヤの編集モードを切り替えることに注意してください。ただしこのテーブルは、airports レイヤの地物を表しています。
- ボタンは、子レイヤ（airports）の編集の全てを保存します。
- ボタンを使用するとマップキャンバス上で airports レイヤのジオメトリのデジタイズを行うことができ、デフォルトで現在の region に新しい地物を割り当てます。アイコンはジオメトリタイプによって変わることに注意してください。
- ボタンは、airports レイヤの属性テーブルに新しいレコードを追加します。この新しい地物はデフォルトで現在の region に割り当てられます。ジオメトリは、 **部分を追加** デジタイジングツールを使用して後から追加できます。
- ボタンは、子レイヤ内の一つまたは複数の子地物を複製することができます。複製した子地物は後から別の親地物に割り当てることもできますし、属性値を編集することもできます。
- ボタンは、選択した空港（複数可）を永久的に削除します。
- シンボルをクリックすると、新しいダイアログが開き、既存の空港を選択して現在の地域に割り当てることができます。これは、誤って違う地域の上に空港を作成してしまった場合に便利です。

- シンボルは、選択した空港（複数可）を現在の地域からリンク解除し、効果的に未割り当て状態（外部キーを NULL に設定）にします。
- ボタンを押すと、選択した子地物にマップをズームさせます。
- 右にある と の 2 つのボタンは、リレーションのある子地物の **テーブル表示とフォーム表示** を切り替えます。

regions レイヤの地物に **ドラッグ＆ドロップデザイナー** を使用している場合には、どのツールを利用可能か選択できます。新しい地物が追加されたときに新しいフォームを開くかどうかは、**地物追加時のフォームを表示しない** オプションで決めることができます。このオプションが正しく動作するためには、NULL でない属性は有効なデフォルト値をとる必要があることに注意してください。

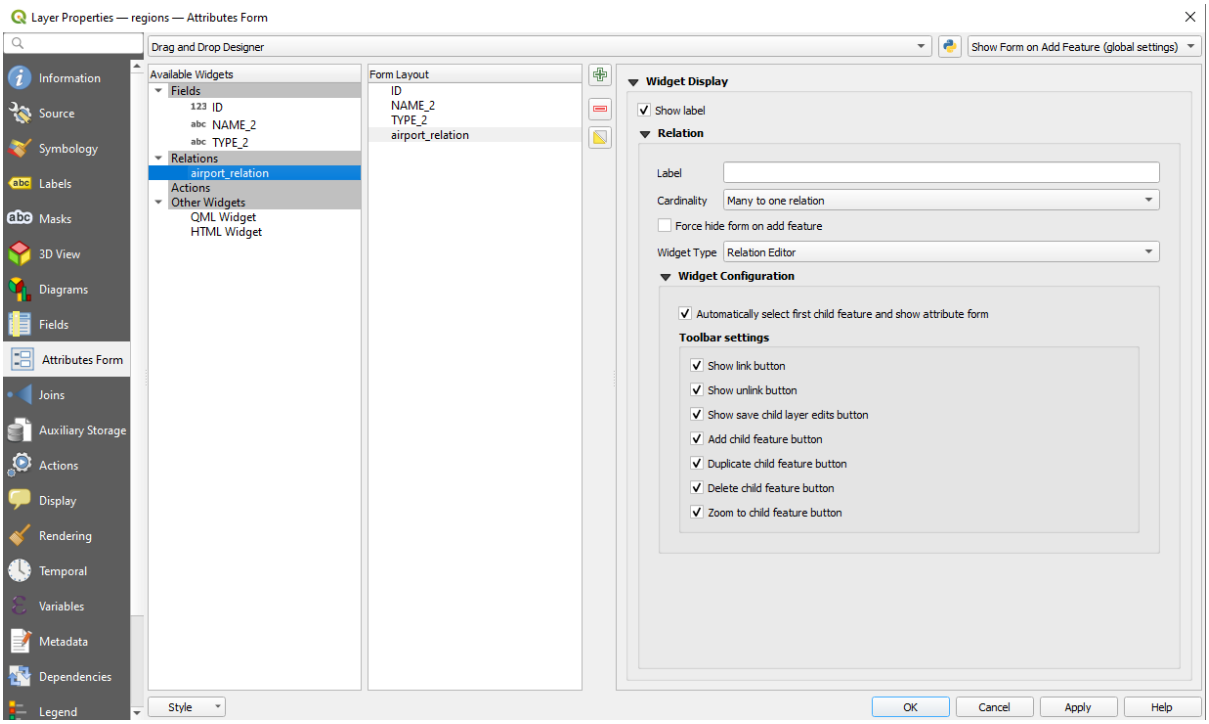


図 16.78: regions レイヤと airports レイヤのリレーションツールを設定するドラッグ＆ドロップデザイナー

上記の例では、参照元レイヤがジオメトリを持っている（つまり、単なる英数字のテーブルではない）ので、上記の手順では、対応するジオメトリ地物を持たないエントリがレイヤの属性テーブルに作成されます。ジオメトリを追加するには、以下の操作を行います。

1. 参照元レイヤの **属性テーブルを開く** を選択します。
2. 被参照レイヤの地物フォーム内で追加したレコードを選択します。
3. 部分を追加 デジタイジングツールを使用して、選択した属性テーブルのレコードとジオメトリを結びつけます。

airports テーブルで作業しているのなら、 `fk_region` フィールド（リレーションを作成するために使用するフィールド）には、「リレーションの参照」ウィジェットが自動的に設定されます。詳細は **リレーションの参照ウィジェット** を参照してください。

airports レイヤのフォームには、 `fk_region` フィールドの右側に ボタンがあることがわかります。こ

のボタンをクリックすると、regions レイヤのフォームが開きます。このウィジェットで、簡単に素早くリンクしている親地物のフォームを開くことができます。

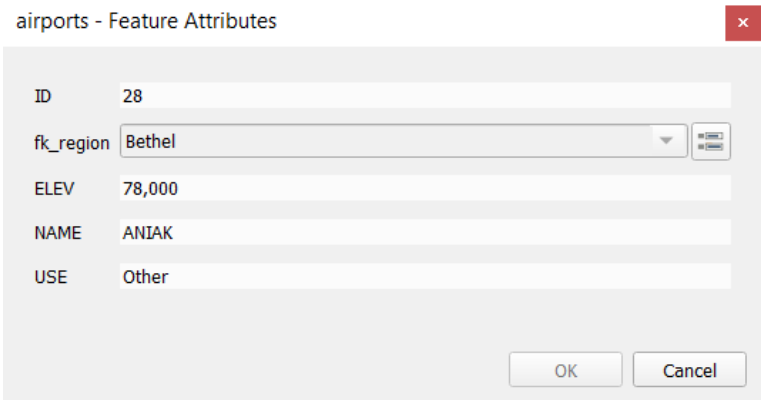
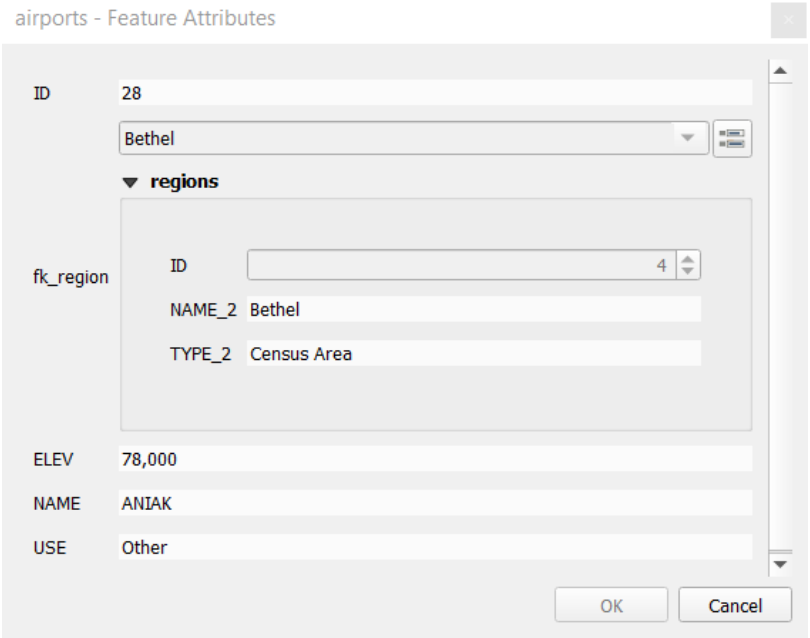


図 16.79: regions へのリレーションを持つ airports の識別ダイアログ

「リレーションの参照」ウィジェットには、親レイヤのフォームを子レイヤに埋め込むオプションもあります。これは、airports レイヤの **プロパティ** **属性フォーム** メニューから設定できます。 `fk_region` フィールドを選択して、 埋め込みフォームの表示 オプションにチェックを入れてください。

今、地物フォームダイアログを見ると、regions のフォームが airports のフォームの中に組み込まれていて、現在の空港を別の地域に割り当てることができるコンボボックスもあることがわかります。



さらに、airports レイヤの編集モードを切り替えると、 `fk_region` フィールドにオートコンプリート機能も追加されます。入力中に regions レイヤの `id` フィールドの値がすべて表示されることがわかります。ここで、airports レイヤの **プロパティ** **属性フォーム** メニューの 新しい地物の追加 オプションを選択した場合、 ボタンを使用して regions レイヤのポリゴンをデジタイズすることができます。

子レイヤは、**値による地物選択** ツールで子レイヤの属性に基づいて親レイヤの地物を選択するために使うこともできます。

図 16.80 では、空港の平均高度が海抜 500m より高い地域すべてを選択しています。

フォームには多数のさまざまな集計関数があることがわかります。

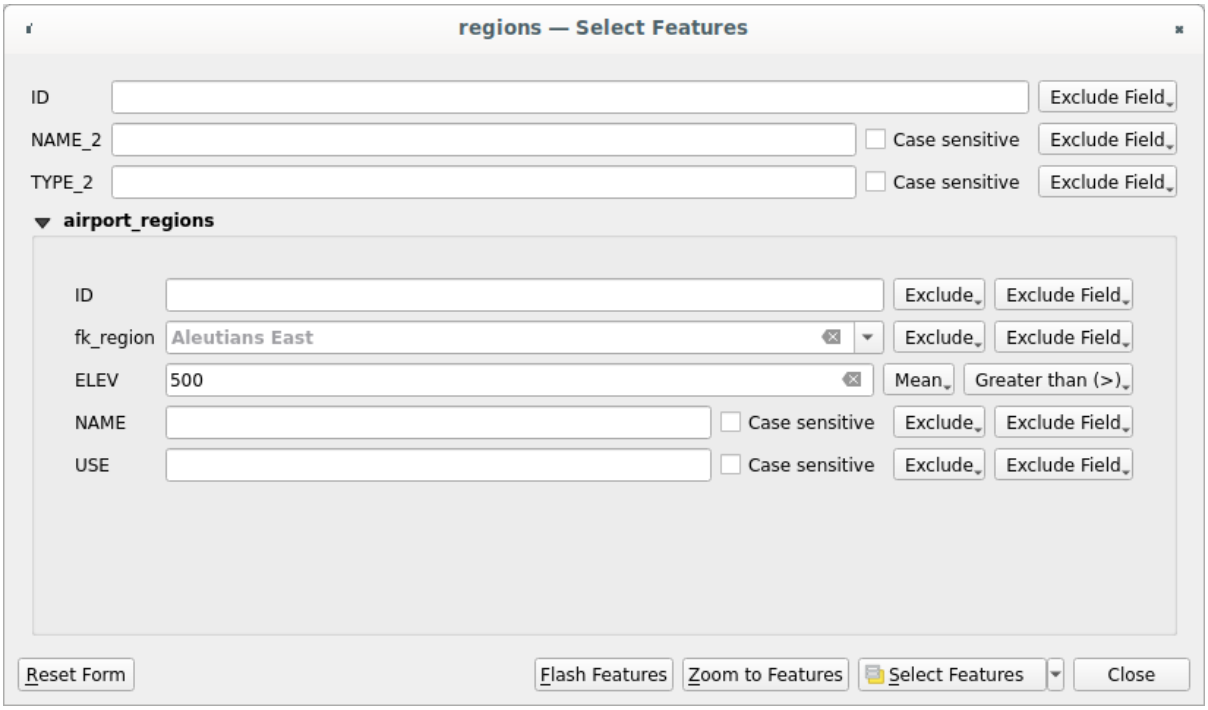


図 16.80: 子レイヤの値で親レイヤの地物を選択する

多対多（N 対 M）リレーションの導入

N 対 M のリレーションとは、2 つのテーブル間の多数対多数の関係のことです。例えば、 airports レイヤと airlines レイヤを考えてみましょう。空港には複数の航空会社が乗り入れており、航空会社は複数の空港に乗り入れています。

以下の SQL コードは、*locations* という名前の PostgreSQL/PostGIS スキーマに、N 対 M リレーションに必要な 3 つのテーブルを作成します。このコードは、PostGIS の **データベース**　*DB* **マネージャ**… や pgAdmin のような外部ツールを使って実行できます。airports テーブルには airports レイヤが、airlines テーブルには airlines レイヤが格納されます。どちらのテーブルにも、わかりやすくするためにいくつかのフィールドが使われています。 **トリッキー** な部分は airports_airlines テーブルです。このテーブルは、すべての空港に対するすべての航空会社（またはその逆）をリストアップするために必要です。この種のテーブルは **ピボットテーブル** として知られています。このテーブルの **制約** は、両方とも互いのレイヤにある場合にのみ、空港と航空会社を関連づけられるものとすることを強制します。

```
CREATE SCHEMA locations;

CREATE TABLE locations.airports
(
   id serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
```

(次のページに続く)

(前のページからの続き)

```
   airport_name text NOT NULL,
   CONSTRAINT airports_pkey PRIMARY KEY (id)
);

CREATE INDEX airports_geom_idx ON locations.airports USING gist (geom);

CREATE TABLE locations.airlines
(
   id serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   airline_name text NOT NULL,
   CONSTRAINT airlines_pkey PRIMARY KEY (id)
);

CREATE INDEX airlines_geom_idx ON locations.airlines USING gist (geom);

CREATE TABLE locations.airports_airlines
(
   id serial NOT NULL,
   airport_fk integer NOT NULL,
   airline_fk integer NOT NULL,
   CONSTRAINT airports_airlines_pkey PRIMARY KEY (id),
   CONSTRAINT airports_airlines_airport_fk_fkey FOREIGN KEY (airport_fk)
      REFERENCES locations.airports (id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
      DEFERRABLE INITIALLY DEFERRED,
   CONSTRAINT airports_airlines_airline_fk_fkey FOREIGN KEY (airline_fk)
      REFERENCES locations.airlines (id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
      DEFERRABLE INITIALLY DEFERRED
 );
```

PostgreSQL の代わりに、GeoPackage を使うこともできます。この場合には、3 つのテーブルは **データベース** *DB* **マネージャ…** を使用して手動で作成します。GeoPackage にはスキーマが無いため、 *locations* の接頭辞は不要です。

`airports_airlines` テーブルの外部キー制約は、 **テーブル** **テーブルを作成…** や **テーブル** **テーブルの編集…** では作成することができません。このため、このテーブルは **データベース** *SQL* **ウィンドウ…** を使用して作成する必要があります。GeoPackage は *ADD CONSTRAINT* 文をサポートしていないため、 `airports_airlines` テーブルは 2 段階で作成する必要があります。：

1. **テーブル** **テーブルを作成…** を使用して、 `id` フィールドのみを持つテーブルを準備します。
2. **データベース** *SQL* **ウィンドウ…** を使用して、以下の SQL コードを入力し実行します：

```
ALTER TABLE airports_airlines
   ADD COLUMN airport_fk INTEGER
   REFERENCES airports (id)
   ON DELETE CASCADE
   ON UPDATE CASCADE
   DEFERRABLE INITIALLY DEFERRED;

ALTER TABLE airports_airlines
   ADD COLUMN airline_fk INTEGER
   REFERENCES airlines (id)
   ON DELETE CASCADE
   ON UPDATE CASCADE
   DEFERRABLE INITIALLY DEFERRED;
```

それから、上で説明したように QGIS で 2 つの *1* **対多のリレーション** を設定します。

- airlines テーブルとピボットテーブルの間のリレーション
- airports テーブルとピボットテーブルとの間の第 2 のリレーション

これを行うためのより簡単な方法（PostgreSQL のみ）は、**プロジェクト　プロパティ　リレーション** の **リレーションを検索** を使用することです。QGIS は自動的にデータベース内の全てのリレーションを読み込むので、必要な 2 つのリレーションを選択するだけで設定ができます。最初に QGIS プロジェクトに 3 つのテーブルを読み込むことを忘れないようにしてください。

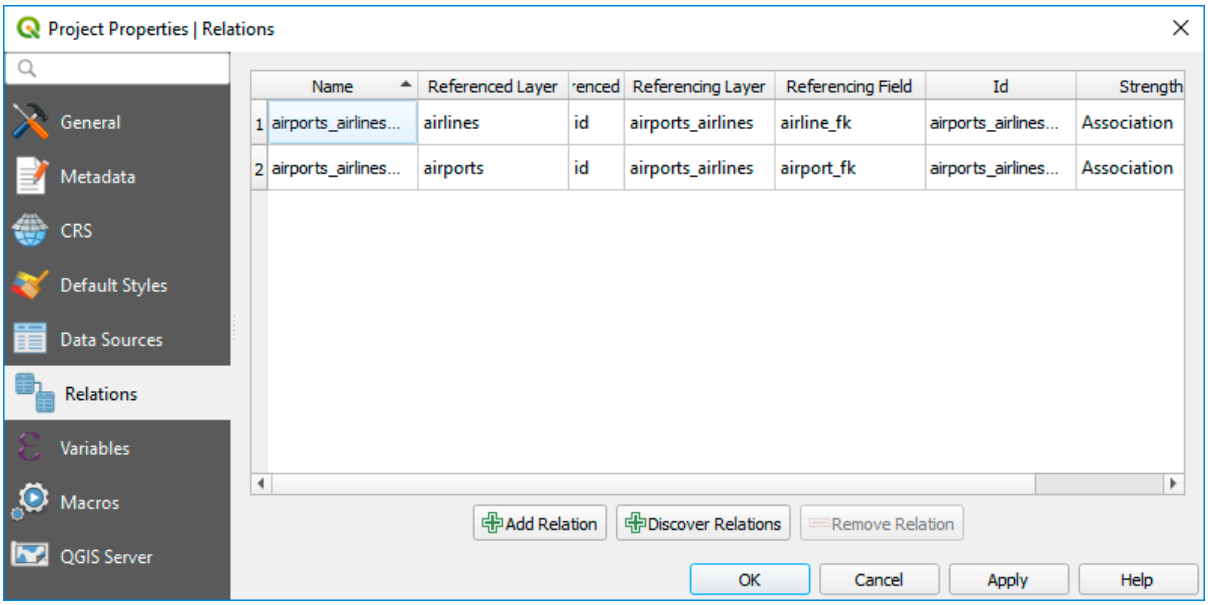


図 16.81: リレーションと自動検索

airport や airline を削除する場合、QGIS は airports_airlines テーブルの関連するレコードを削除しません。この作業は、現在の例のようにピボットテーブルの作成時に正しい **制約** を指定すれば、データベースによって行われます。

注釈: N 対 M のリレーションと自動トランザクショングループを組み合わせる

このようなコンテキストで作業する場合は、 **プロジェクトのプロパティ　データソース** でトランザクションモードを有効にする必要があります。 QGIS は、すべてのテーブル（航空会社、空港、ピボットテーブル）の行を追加または更新できる必要があります。

最後に、**レイヤプロパティ　属性フォーム** で、`airports` レイヤと `airlines` レイヤに対して正しい「要素数」を選択する必要があります。最初のレイヤには **airlines (id)** オプションを、2 番目のレイヤには **airports (id)** オプションを選択する必要があります。

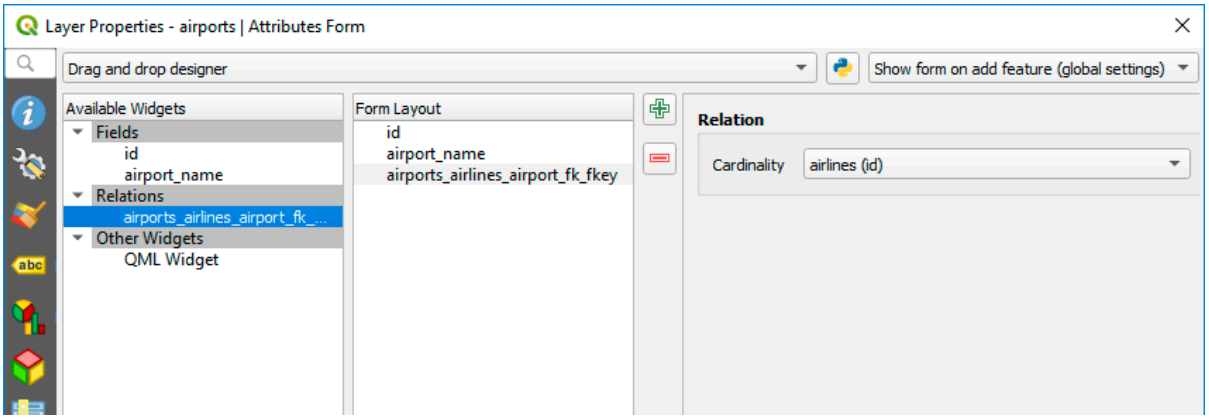


図 16.82: 関係の要素数の設定

これで、サブフォーム内で **子地物の追加** や **既存の子地物をリンク** を使用して airline に airport を（そして airport に airline を）関連付けることができるようになりました。 `airports_airlines` テーブルにはレコードが自動的に挿入されます。

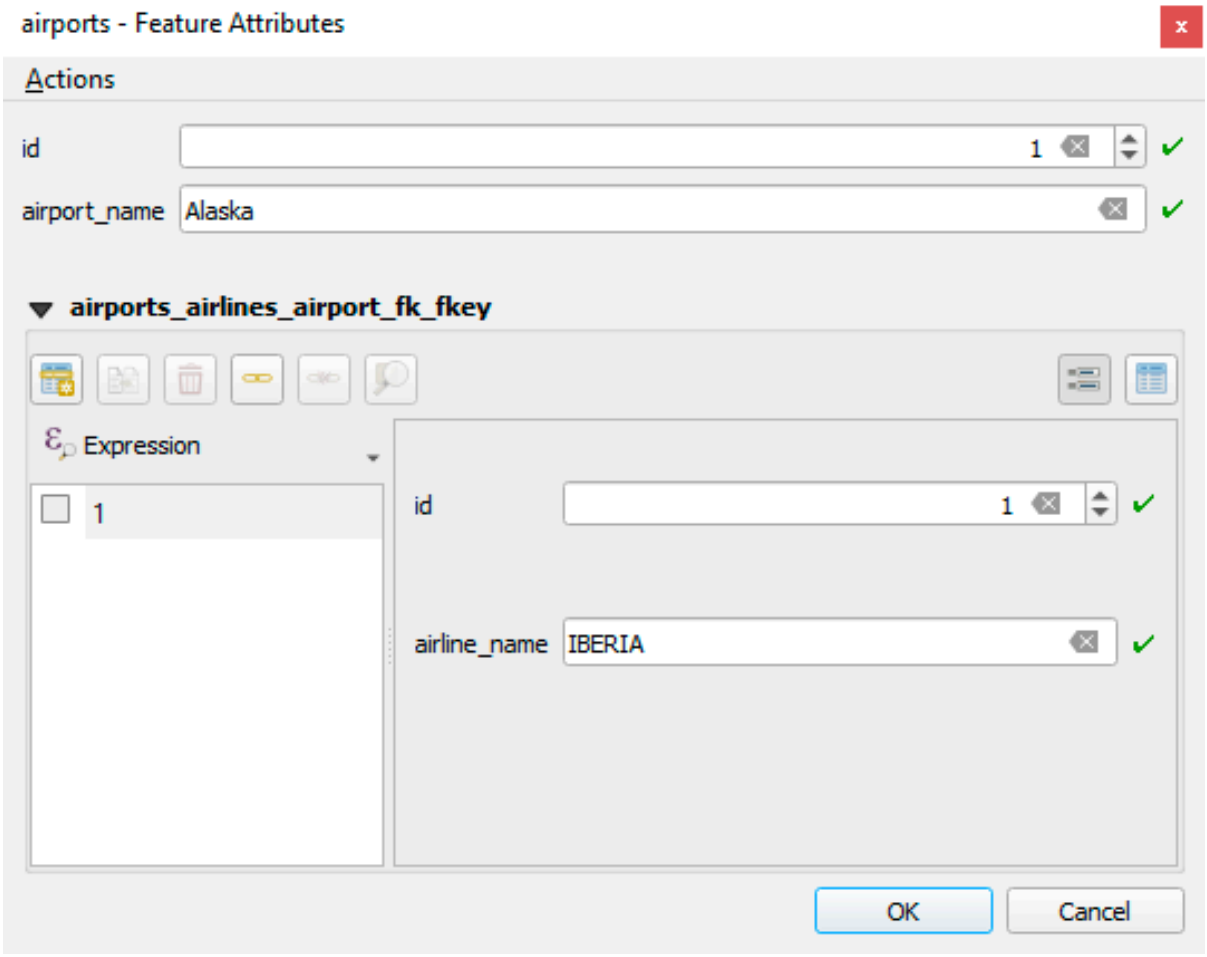


図 16.83: airports と airlines の N 対 M リレーション

注釈: 多対 1 リレーション 要素数を使用する

N 対 M リレーションでピボットテーブルを隠すことが望ましくない場合があります。これは主に、リレーションが確立されたときにのみ値を持つことができる属性がリレーションに存在するというのが理由です。テーブルが（ジオメトリフィールドを持つ）レイヤの場合、ピボットテーブルの外部キーフィールドで **地図上の地物特定** オプション（ **レイヤプロパティ　属性フォーム　利用可能なウィジェット　フィールド** ）を有効にするとよいかもしれません。

注釈: ピボットテーブルの主キー

ピボットテーブルの主キーには複数のフィールドを使用しないでください。QGIS は単一の主キーを想定しているため、 `constraint airports_airlines_pkey primary key (airport_fk, airline_fk)` のような制約は機能しません。

多態リレーションの導入

多態リレーションは 1-N 関係の特殊なケースであり、1 つの参照元（ドキュメント）レイヤ内に複数の参照先レイヤの地物が含まれます。これは、参照先レイヤごとに異なる参照元レイヤを必要とする通常のリレーションとは異なっています。参照元（ドキュメント）レイヤに `layer_field` カラムを追加することで単一の参照元（ドキュメント）レイヤとすることができ、ここには参照先レイヤを識別する情報を格納します。最も単純な形は、参照元（ドキュメント）レイヤはこのフィールドに参照先レイヤのレイヤ名を入れるだけです。

より正確に言うならば、多態リレーションは、参照元レイヤが同じ通常のリレーションの集合ですが、参照先レイヤは動的に定義されるものです。レイヤの多態リレーションの設定は、テーブル名、レイヤ ID、レイヤ名など、参照先レイヤのいくつかのプロパティと一致しなければならない式を使用することで解決されます。

公園に行き、そこで見かけたさまざまな種類の植物 `plants` や動物 `animals` の写真を撮ることを考えてみましょう。それぞれの植物や動物には複数の写真が関連付けられているので、通常の 1:N リレーションで写真を保存する場合、 `animal_images` と `plant_images` という 2 つの別々のテーブルが必要です。テーブル 2 つならば問題とはならないかもしれませんが、キノコや鳥などの写真も別々に撮影したいとしたらどうでしょうか。

多態リレーションは、参照元のすべての地物を同一の `documents` テーブルに格納することでこの問題を解決します。各地物の `referenced_layer` フィールドには参照先レイヤが、 `referenced_fk` フィールドには参照先の地物 ID が格納されます。

多態リレーションの定義

最初に、QGIS にレイヤ間の多態リレーションについて知らせましょう。これは、 **プロジェクト** **プロパティ...** から行えます。 **リレーション** タブを開き、 **リレーションを追加** ボタンの横にある小さな下向き矢印のボタンをクリックします。新しく現れたドロップダウンメニューから、 **多態リレーションを追加** オプションを選択します。

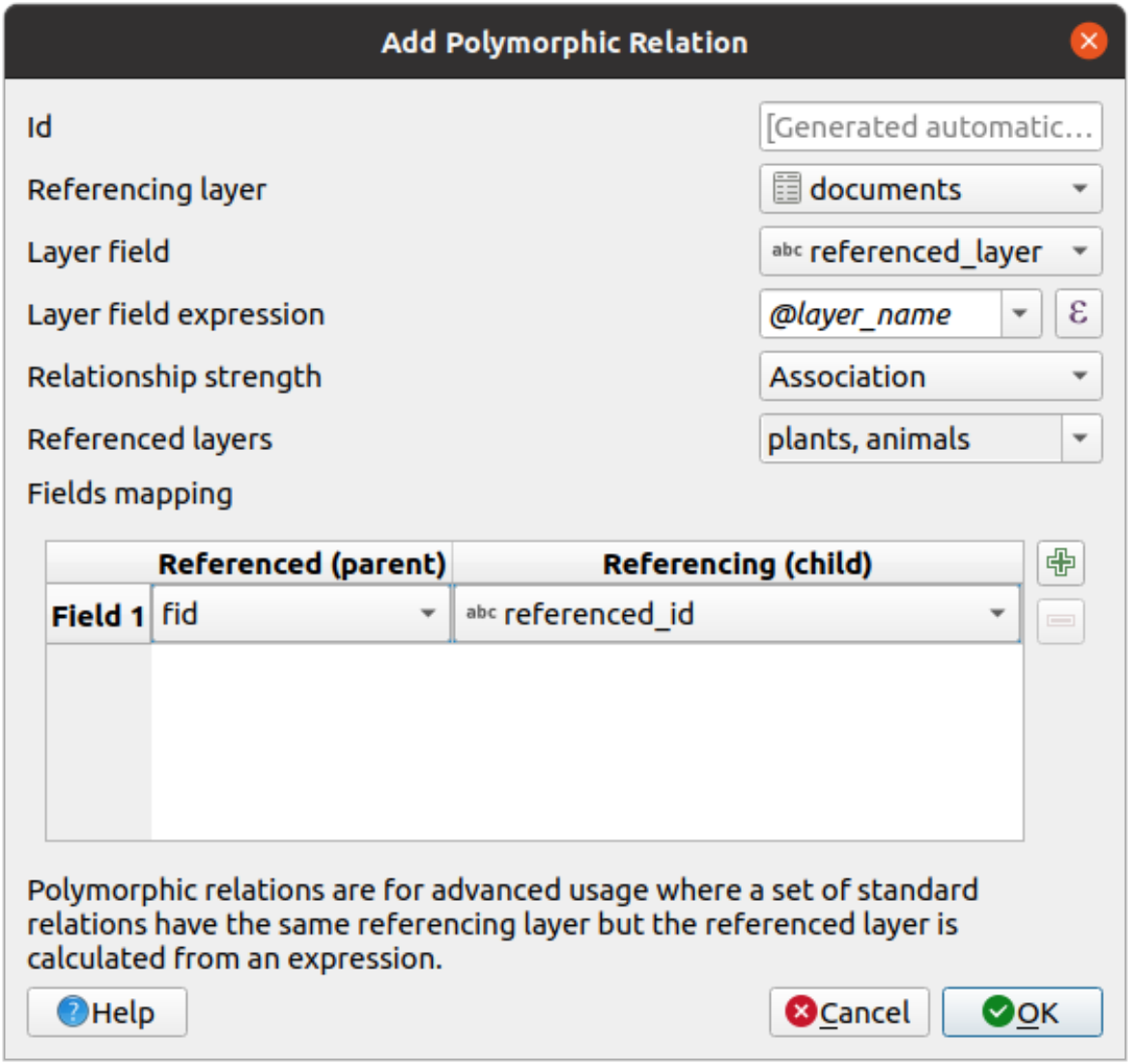


図 16.84: 参照元レイヤとして documents を、参照先レイヤとして animals と plants を使用した多態リレーションを追加します。

- **Id** は内部的な目的で使用され、ユニークでなければなりません。これは **カスタムフォーム** を作成するために必要となる場合があります。空のままにしておくと自動で Id を作成しますが、扱いやすいように自分で Id を指定することもできます。
- **参照元レイヤ（子）** は子レイヤとも呼ばれ、外部キーのフィールドを持っているレイヤです。この例では documents レイヤです。このレイヤには他のレイヤを指す参照フィールドを追加する必要があり、これが referenced_fk です。

注釈: 場合によっては、レイヤ内の地物を一意に識別するためには複数のフィールドが必要となることがあります。このようなレイヤとのリレーションを作成するためには、**複合キー** 、すなわち、一致するフィールドの複数ペアが必要です。 複合外部キーとしてペアのフィールドを追加 ボタンを使用して、必要な数のペアを追加できます。

- **レイヤのフィールド** は、評価されたレイヤ式の結果を格納する参照元テーブルのフィールドです。

これは、この地物がどのレイヤに属するかの参照テーブルです。 この例では、 `referenced_layer` フィールドがこれに該当します。

- **レイヤのフィールド式** は、レイヤのユニークな識別子を評価します。これは、レイヤ名 `@layer_name` やレイヤの ID `@layer_id` 、レイヤのテーブル名 `decode_uri(@layer, 'table')` など、レイヤを一意に識別できるものであれば何でも利用できます。
- **リレーションの強度** は、親子レイヤ間に生成されたリレーションの強さを設定します。デフォルトの *Association* タイプは、親レイヤが子レイヤに **単に** リンクされていることを意味します。 一方、 **コンポジション** タイプでは、親レイヤを複製するときに子レイヤも複製され、親レイヤの地物を削除すると子レイヤの地物も削除されます。これはすべてのレベルにカスケードしていきます（つまり、子レイヤのそのまた子レイヤ... も一緒に削除されます）。
- **参照先レイヤ** は親レイヤとも呼ばれ、主キーを持ち、指し示される側のレイヤです。この例では、`plants` レイヤと `animals` レイヤがこれに該当します。参照先レイヤの主キーをドロップダウンメニューから定義する必要があり、この例では `fid` です。有効な主キーを定義するには、全ての参照先レイヤで同じ名前のフィールドが存在する必要があることに注意してください。そのようなフィールドが存在しない場合には、多態リレーションを保存することができません。

一度追加したら、この多態リレーションは **多態リレーションを編集** メニューエントリから編集ができます。

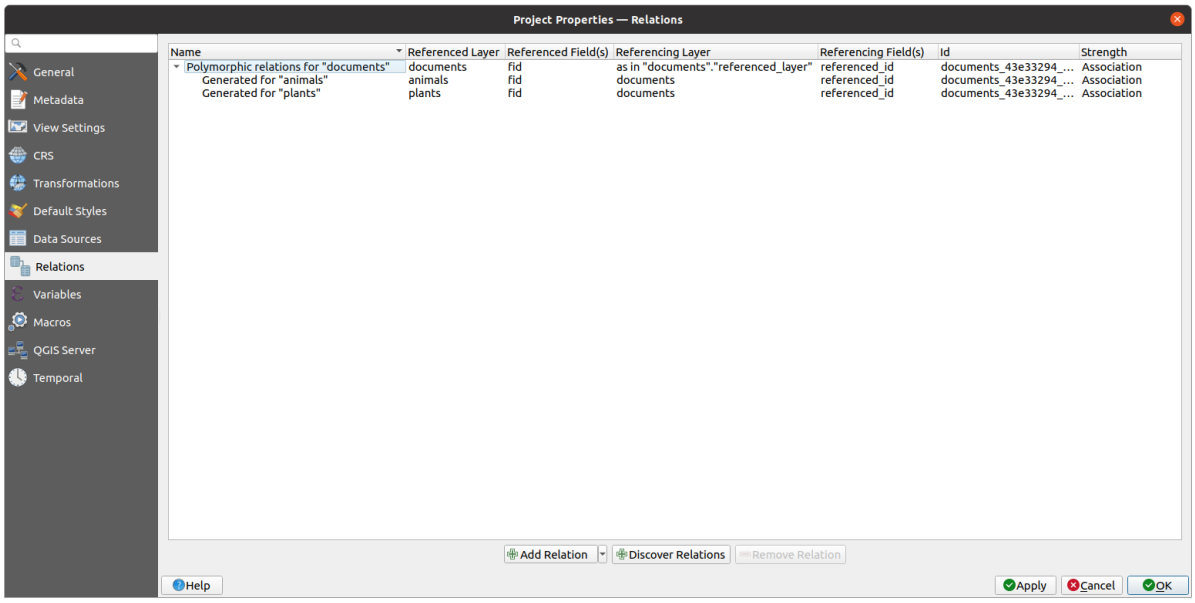


図 16.85: 新規作成した多態リレーションと animals と plants に関する子リレーションのプレビュー

上記の例では、以下のデータベーススキーマを使用しています：

```
CREATE SCHEMA park;

CREATE TABLE park.animals
(
   fid serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   animal_species text NOT NULL,
```

(次のページに続く)

(前のページからの続き)

```
   CONSTRAINT animals_pkey PRIMARY KEY (fid)
);

CREATE INDEX animals_geom_idx ON park.animals USING gist (geom);

CREATE TABLE park.plants
(
   fid serial NOT NULL,
   geom geometry(Point, 4326) NOT NULL,
   plant_species text NOT NULL,
   CONSTRAINT plants_pkey PRIMARY KEY (fid)
);

CREATE INDEX plants_geom_idx ON park.plants USING gist (geom);

CREATE TABLE park.documents
(
   fid serial NOT NULL,
   referenced_layer text NOT NULL,
   referenced_fk integer NOT NULL,
   image_filename text NOT NULL,
   CONSTRAINT documents_pkey PRIMARY KEY (fid)
);
```

16.2.7 外部リソースの保存と取得

フィールドは外部ストレージシステムに保存されたリソースを対象とすることもできます。属性フォームは外部ストレージシステムに対するクライアントとして動作するように設定でき、フォームから直接、ユーザーの要求に応じて外部ストレージシステムからリソースを保存・取得することができます。

外部ストレージの設定

外部ストレージを設定するにはまず、最初にベクタレイヤの **属性フォームプロパティ** を設定し、 **アタッチメント** ウィジェットを選択する必要があります。

図 16.86: 指定したフィールドの WebDAV 外部ストレージの設定編集

アタッチメント ウィジェットを選択したら、まずは **ストレージタイプ** を以下の中から選択します：

- **ファイルを選択** ：ターゲットの URL がある場合に使用します。リソースを選択すると、ストア操作は行われず、属性値は単に URL で更新されます。
- **単純コピー** ：リソースのコピーをファイルディスクの指定位置（ローカル、ネットワーク共有ファイルシステムどちらも可）に保存し、属性値はコピーへのパスで更新されます。
- *WebDAV Storage*： リソースは WebDAV プロトコルをサポートする HTTP サーバーにプッシュされ、属性はその URL で更新されます。Nextcloud や Pydio などのファイルホスティングソフトウェアがこのプロトコルに対応しています。

続いて、 **保存** *URL* パラメータを設定する必要があります。これは、新しいリソースを保存する必要がある場合に使用する URL です。これは地物の属性に応じた特定の値を持てるように、 **データによって定義された上書きウィジェット** を使用して式を設定することもできます。

変数 **@selected_file_path** を式中で使用すれることができ、ユーザーが（ファイルセレクタを使用して、あるいはドラッグ＆ドロップで）選択したファイルの絶対ファイルパスを表すことができます。

注釈: **WebDAV** 外部ストレージを使用しており URL が "/" で終わっている場合には、これはフォルダとみなされ、選択したファイルの名前が後ろに追加されて最終的な URL となります。

外部ストレージシステムが必要とするならば、 **認証** に関する設定も行えます。

外部ストレージの利用

設定が完了したら、地物の編集中状態で ... ボタンを押すことで、ローカルファイルを選択できます。 **ストレージタイプ** に応じてこのファイルは外部ストレージシステムに保存され（ **ファイルを選択** を選択した場合を除く）、フィールドの値は新しいリソースの URL で更新されます。

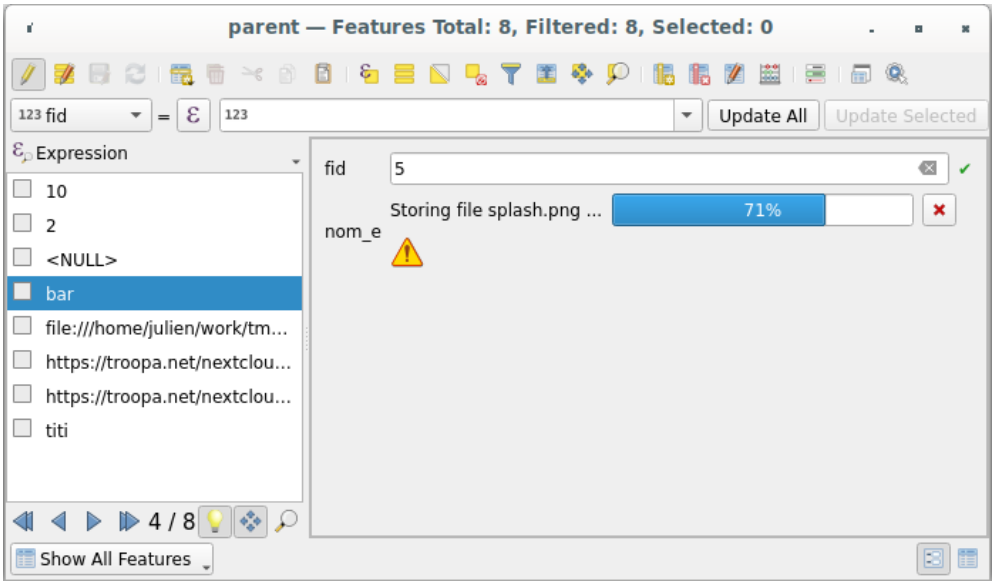


図 16.87: ファイルの WebDAV 外部ストレージへの保存

注釈: ファイルアタッチメントウィジェット上にファイルをドラッグ&ドロップすることでも、同様の結果が得られます。

キャンセル ボタンを使用すると、保存プロセスが中断します。 **統合ドキュメントビューア** を使用してビューアの設定を行うことで、外部ストレージシステムからリソースを自動的に取得し、URL の下に表示させることができます。上記の アイコンは、外部ストレージシステムからリソースを取得できないことを示しています。この場合、より詳細な情報が **ログメッセージパネル** に表示されていることがあります。

16.3 編集

QGIS は、OGR、SpatiaLite、PostGIS、MS SQL Server、Oracle Spatial のベクタレイヤやテーブルを編集する様々な機能を備えています。これらは、2D または 3D ジオメトリタイプのものです。

注釈: GRASS レイヤを編集する手順は異なります。詳細は *GRASS* **ベクタレイヤをデジタイズして編集する** のセクションを参照してください。

注意: 同時編集

QGIS は、他の人があなたと同時に同じ地物を編集していても追跡しません。最後に編集を保存した人が勝ちます。

Tip: 編集の妥当性検証

レイヤ プロパティ **デジタイジング** タブの設定で、連続的な妥当性検証をレイヤ単位で有効にすることができます。詳細については **デジタイズプロパティ** を参照してください。

16.3.1 スナップ許容範囲と検索半径の設定

設定 **オプション...** **デジタイズ** メニューには、QGIS の編集ツールのデフォルトの動作を設定するためのパラメータがいくつかあります。詳細な情報は **デジタイズの設定** を参照してください。

ベクタレイヤのジオメトリを最適かつ正確に編集するためには、地物の頂点のスナップ許容量と検索半径を適切な値に設定する必要があります。 **スナップ** グループはこれに関連する、スナップ許容量と検索半径の扱いに関するオプションを提供します。

- スナップ許容量：新しい頂点を追加したり既存の頂点を移動するとき、接続しようとしている頂点またはセグメントに最も近いものを検索するために QGIS が使用する距離がスナップ許容量です。カーソルがスナップ許容範囲内にない場合は、QGIS は既存の頂点またはセグメントにスナップするのではなく、マウスボタンを離した場所に頂点を落とします。

  この許容量の設定はスナップを行うすべてのツールに影響し、新しいレイヤやプロジェクトにデフォルトとして適用されます。ただしこの設定は、レイヤレベルで上書きすることができます（ **スナップとデジタイズのオプション** 参照）。

- 検索半径： **頂点編集用検索半径** は、地図上をクリックしたときに選択する頂点を 検索 するために QGIS が使用する距離です。カーソルが検索半径内にいない場合、QGIS は編集のために頂点を見つけ選択することはありません。

スナップ許容量や検索半径は 地図上の単位 または ピクセル で設定します。ちょうど良い値を設定するには試行錯誤する必要があるかもしれません。大きすぎる値を設定した場合、特に近接した多数の頂点を扱う場合には、QGIS は間違った頂点にスナップすることがあります。検索半径を小さくすると、動かしたい頂点にヒットさせるのが難しくなります。

16.3.2 スナップとデジタイズのオプション

グローバルな *Snapping and Digitizing Settings* (スナップモード、許容値、単位...) は、プロジェクト内で **プロジェクト　スナップオプション...** `**メニューからオーバーライドすることができます。** *:guilabel:`Snapping and Digitizing Options* では、その他のプロパティ（スナップレイヤー、スケールリミット、トポロジ...）も設定することができます。

デフォルトでは、 スナップを有効にする ボタンを押すかキーボードの S を押すまでは、プロジェクトにおいてスナップは無効化されています。スナップモード、スナップ許容量の値と単位はスナップツールバーでも設定できます。

スナップのプロパティ

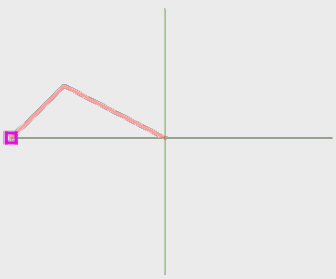
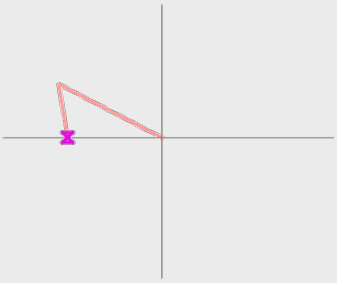
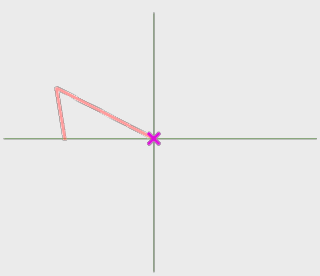
レイヤへのスナップには、3 種類の選択肢があります：

- **すべてのレイヤ** ：これはプロジェクト内のすべての可視状態のレイヤに対するクイック設定で、ポインターがすべての頂点やセグメントにスナップします。多くの場合、このスナップモードを使用すれば十分ですが、多数のベクタレイヤがあるプロジェクトでこのモードを使用する場合には注意が必要です。パフォーマンスに影響を及ぼすことがあります。
- **アクティブレイヤ** ：スナップに使用するのはアクティブレイヤのみです。編集中のレイヤ内でトポロジの一貫性を確保するのに便利な方法です。
- **詳細設定** ：レイヤ単位でスナップモードの有効化や許容量とその単位、重なりの扱いや縮尺範囲を調整できます（ 図 16.88 参照）。あるレイヤの編集で、その頂点を他のレイヤにスナップさせたい場合、ターゲットとなるレイヤにチェックを入れ、スナップ許容量に大きな値を入れるようにしてください。スナップオプションダイアログでチェックが入っていないレイヤに対しては、スナップが発生しません。

スナップモードについては、 頂点 、 セグメント 、 領域（Area） 、 重心点 、 セグメントの中央 、 線のエンドポイント の中から選択できます。

QGIS では **スナップ** の種類に応じてさまざまな **スナップ** アイコンを表示します：

表 16.2: スナップのアイコン

		
頂点へのスナップ：四角形アイコン	セグメントへのスナップ：砂時計型アイコン	交点へのスナップ：バツ印アイコン

グローバル設定の **デジタイズ** において、これらのアイコンの色を変更できます。

スナップ許容量は、プロジェクトの 地図上の単位 または ピクセル のいずれかで設定できます。 ピクセルを選択する利点は、さまざまなマップの縮尺でスナップ許容量を一定に保つことができることです。通常は 10 から 12 ピクセルが適当な値ですが、ディスプレイの DPI にもよります。地図上の単位を使用すると、許容量を実際の地上距離と関連付けることができます。例えば、要素間の最小距離の設定がある場合には、このオプションはお互いに近すぎる頂点を追加しないようにするのに役立ちます。

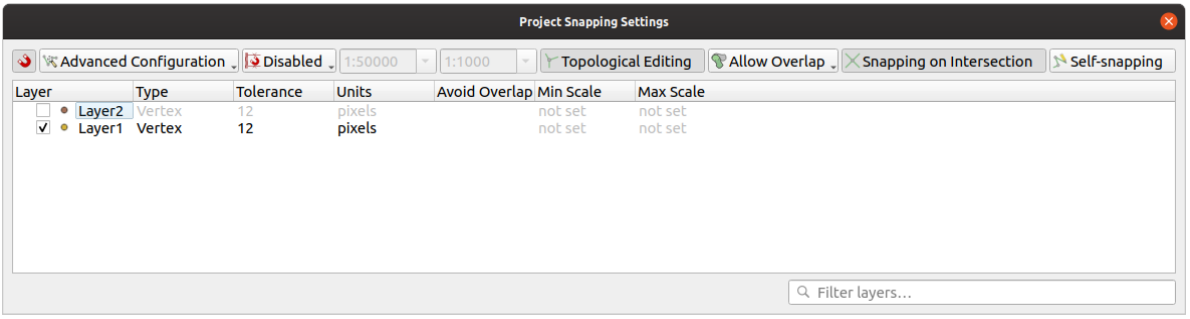


図 16.88: スナップオプション（詳細設定モード）

注釈: デフォルトでは、可視状態の地物（スタイルが表示されている地物、ただしシンボロジが "シンボルなし" のレイヤを除く）のみにスナップできます。非表示の地物にもスナップできるようにするには、 **設定　オプション　デジタイズ** タブで **非表示の地物に対してもスナップを有効にする** にチェックを入れます。

Tip: デフォルトでスナップを有効にする

設定　オプション　デジタイズ タブで、すべての新規プロジェクトでスナップをデフォルトで有効に設定することができます。また、既定のスナップモードや許容量、単位を設定することもでき、この設定値が **スナップオプション** ダイアログに入力されます。

交点でのスナップを有効にする

このほかに利用可能なオプションとして、 **交点でのスナップを有効にします** の使用があります。これは、交差部に頂点が無くても、スナップが有効なジオメトリの交点にスナップできるようになります。

スナップのスケール範囲制限

場合によっては、スナップが非常に遅くなることがあります。これは、いくつかのレイヤ内に地物が多いことが原因で、計算と管理に重いインデックスを必要とするためです。スナップの有効化をマップビューが関連するスケール範囲内にあるときのみとするパラメータがあります。これにより、スナップに関連したコストのかかるインデックス計算は、関連するスケールでのみ行うようにすることができます。

スナップのスケール制限は、 **プロジェクト　スナップオプション...** で設定ができます。スナップのスケール制限は、 **詳細設定** モードでのみ利用可能です。

スナップをスケール範囲で制限するには、3 つのモードが利用できます：

- **無効化** ：スナップはどのようなスケールでも有効です。これがデフォルトのモードです。
- **グローバル** ：スナップは制限され、現在の縮尺がグローバルな最小値と最大値の間にあるときのみ有効となります。このモードを選択した場合は、利用可能となる 2 つのウィジェットでスナップが有効となる縮尺範囲を設定します。
- **レイヤ単位** ：スナップ縮尺範囲の制限は各レイヤで定義します。このモードを選択すると 2 つの列が利用可能となり、各レイヤの最小縮尺と最大縮尺を設定できます。

最小スケールと最大スケールは QGIS の規則に従うことに注意してください。最小スケールは最も「ズームアウト」した、最大スケールは最も「ズームイン」したスケールです。最小スケールや最大スケールが「0」または「未設定」に設定されている場合、制限は無いものとみなされます。

自己スナップ

オプションを使用すると、現在編集中のジオメトリにスナップできるようになります。 **高度なデジタイズパネル** と組み合わせると、先に描画した辺や頂点に相対的な新しい辺をデジタイズする便利な方法を提供します。自己スナップは無効なジオメトリを引き起こす場合があるため、注意して使用してください。

カスタムグリッドへのスナップ

スナップ距離は、レイヤプロパティダイアログの **デジタイズ** タブでレイヤ単位でもカスタマイズできます。 **ジオメトリの精度** の距離を設定することで、マップキャンバスに表示できる縮尺の場合には点のグリッドが表示されるようになります。このとき、スナップは点のグリッドに対して行われ、追加または変更されたジオメトリは、そのすべての頂点が最も近いグリッド点にスナップされます。詳細については、 **デジタイズプロパティ** を参照してください。

16.3.3 トポロジ編集

これらのスナップオプションに加えて、 **スナップオプション**... ダイアログ（ **プロジェクト** → **スナップオプション** ）と **スナップ** ツールバーでは、トポロジ編集機能を有効 / 無効にすることができます。

トポロジ編集を有効にする

ボタンは、共通の境界を持つ地物を編集および維持する際に役立ちます。このオプションを有効にすると、QGIS は共有された境界を「検出」します。共通の頂点/セグメントを移動すると、QGIS は隣接する地物のジオメトリの頂点やセグメントも移動させます。

トポロジ編集は、レイヤが表示されていて編集モードであれば、別のレイヤの地物でも機能します。

Z 値や M 値のあるレイヤーでは、トポロジ編集で、接続に使用したエッジの値に基づいて、頂点の Z 値や M 値を補間します。

重なりの制御

重なりの防止によって、選択されたレイヤの既存の地物と重なるような新しい地物の作成を防ぐことができるため、隣接するポリゴンのデジタイズが高速化できます。これは、重なりツールでコントロールすることができます。以下に挙げる 3 つのモードが利用可能です：

1. **重なりを許容する**（デフォルト）
2. **アクティブレイヤの重なりを禁止**：編集中のレイヤの他の地物との重なりを防ぎます。新しいジオメトリを隣接するジオメトリと重なるようにデジタイズすると、QGIS は新しいジオメトリの重なる部分を切り取り、既存の地物の境界線にスナップさせます。境界上の共通の頂点をデジタイズする必要がないことが利点です。
3. **詳細設定に従う**：重なりの設定は **詳細設定** モードでレイヤ毎に設定できます。

注釈: 新しいジオメトリが既存のポリゴンによって完全に覆われている場合には、新しい地物は消去され、QGIS はエラーメッセージを表示します。

警告: 重なりを避ける オプションは注意して使用すること

このオプションは任意のポリゴンレイヤの重なりのある新しい地物をカットするため、必要なくなったときにこのオプションのチェックを外し忘れていると、思いがけないジオメトリが作成されることがあります。

自動トレース

通常、地図キャプチャツール（地物を追加、部分を追加、リングを追加、地物の変形、分割）を使用する際には、地物の各頂点をクリックする必要があります。自動トレースモードを使えば、デジタイズ時にすべての頂点を手作業で配置する必要がなくなるため、デジタイズ作業をスピードアップできます。

1. アイコンをクリックするか T キーを押して、トレースを有効にします ツールを有効にします（**スナップ** ツールバー内）。
2. トレースしたい地物の頂点またはセグメントに **スナップ** させます。
3. スナップさせたい別の頂点やセグメント上にマウスを移動させると、通常の直線ではなく、デジタイズのラバーバンドが最後にスナップした点から現在の位置までのパスを表現します。このツールは曲線ジオメトリにも対応しています。

   QGIS は、2 つのポイント間の最短パスを構築するためにトレース元の地物のトポロジを実際に使用します。トレースの際には、パスを構築するためにトレース元のレイヤでスナップを有効にする必要があります。また、デジタイズ中に既存の頂点やセグメントにスナップし、2 つのノードが既存の地物の辺を介してトポロジ的に接続可能であることを保証する必要があります。そうでない場合、QGIS はノードを接続することができず、単一の直線をトレースします。

4. 別の頂点やセグメント上でクリックすると、QGIS は表示されたパスに沿って途中の頂点を配置します。

アイコン横のメニューを展開して *Offset* オプションを設定すると、地物に沿ってトレースするのではなく、地物に平行なパスをデジタイズします。正の値は新しい線がトレース方向に対して左側に移動し、負の値は右側に移動します。

注釈: 最適なトレースのための地図縮尺やスナップ設定の調整

地図表示内にあまりにも多くの地物がある場合、トレース構造の準備に長時間かかり、メモリのオーバーヘッドが大きくなる可能性を回避するため、トレースは無効になります。ズームインするか、いくつかのレイヤを非表示にすると、トレースは再び有効になります。

注釈: トポロジ点は追加されません

たとえ **トポロジ編集** が有効になっているとしても、自動トレースツールは既存のポリゴンジオメトリに点を追加しません。編集中のレイヤでジオメトリの精度が有効になっている場合、結果として得られるジオメトリは既存のジオメトリを正確にトレースしていない場合があります。

Tip: T キーを押して自動トレースを素早く有効・無効にする

T キーを押すと、いつでも（地物のデジタイズ中でも）トレースを有効化/無効化することができます。従って、地物の一部分はトレースを有効にして、残る部分はトレースを無効にしてデジタイズすることが可能です。トレースが無効の場合には、スナップツールは通常どおりに動作します。

Tip: トレースを曲線ジオメトリに変換する

設定 **オプション** **デジタイズ** **トレース** の設定で、デジタイズで曲線ジオメトリを作成することが可能になります。詳細については **デジタイズオプション** を参照してください。

16.3.4 既存レイヤのデジタイズ

デフォルトでは、QGIS はレイヤを読み取り専用でロードします。これは、マウスの操作を誤ってレイヤを編集してしまうことを防ぐための安全装置です。ただし、データプロバイダが編集をサポートしており（ **データ形式とフィールドを探究する** 参照）、かつ基礎となるデータソースが書き込み可能である（つまり、そのファイルが読み取り専用ではない）場合に限り、レイヤの編集が選択できます。

Tip: プロジェクト内のレイヤの編集権限を制限する

プロジェクト **プロパティ...** **データソース** **レイヤの** *Capabilities* テーブルでは、データプロバイダの権限とは無関係に、任意のレイヤを読み取り専用に設定することができます。これは、マルチユーザー環境において権限のないユーザーが誤ってレイヤ（シェープファイル等）を編集し、データが壊れてしま

うことを防ぐための簡便な方法です。この設定は、現在のプロジェクト内でのみ適用されることに注意してください。

ベクタレイヤを編集するツールは、「デジタイジングツールバー」と、 **高度なデジタイズ** セクションで説明する「高度なデジタイズツールバー」の二つに分かれています。どちらも **ビュー** **ツールバー** で選択と解除ができます。

基本のデジタイズツールには、以下の機能があります：

表 16.3: ベクタレイヤの基本的な編集ツールバー

ツール	目的	ツール	目的
	全レイヤまたは選択レイヤを同時に保存・ロールバック・編集キャンセルするツールへのアクセス		アクティブなレイヤのステータスに基づいて、選択したレイヤ（複数可）の編集ステータスのオン・オフを切り替え
	アクティブレイヤの編集内容を保存		
	直線のセグメントでデジタイズする		曲がった線を使ってデジタイズする
	フリーハンドによるデジタイズを有効にする		正多角形のポリゴンをデジタイズする
	レコードを追加		地物を追加：ポイントをキャプチャ
	地物を追加：ラインをキャプチャ		地物を追加：ポリゴンをキャプチャ
	頂点ツール（全レイヤ）		頂点ツール（現在のレイヤ）
 **頂点エディタを表示**	頂点エディタパネルを自動的に開くかどうかを設定する		全選択地物の属性一括変更
	選択した地物をアクティブレイヤから削除		地物をアクティブレイヤから切り取り
	選択した地物をアクティブレイヤからコピー		地物をアクティブレイヤに貼り付け
	アクティブレイヤの変更を元に戻す		アクティブレイヤの変更をやり直す

いずれのデジタイジングツールでも、使用中にツールへのフォーカスを失うことなく、マップキャンバス内で **ズームやパン** を実行できます。

編集セッションはすべて、  編集モード切替 オプションを選択することで開始します。このオプションは、レイヤのコンテキストメニューや属性テーブルダイアログ、デジタイジングツールバー、 **レイヤ** メニューにあります。

レイヤが編集モードになると、編集ツールバーのツールボタンが追加で利用可能になり、すべての地物の頂点にマーカーが表示されます。マーカーは、 **設定** **オプション...** **デジタイズ** メニューの **選択地物のみマーカーを表示する** オプションがチェックされていない限り、すべての地物の頂点に表示されます。

Tip: 定期的に保存しましょう

定期的に  レイヤ編集内容の保存 で保存するのを忘れないでください。また、これはデータソースが変更をすべて受け入れ可能かどうかをチェックします。

ジオメトリを編集するテクニック

ジオメトリ描画ツール（主に地物の追加、分割、再形成を行うもの）がライン又はポリゴン型のレイヤで有効になっている場合、新しい頂点を追加する手法を選択することができます:

-  セグメントでデジタイズ: 左クリックで開始点と終了点が定義された直線状のセグメントを描画します。
-  曲線でデジタイズ: 左クリックで定義された 3 つの連続したノード（開始、弧に沿った点、終了）に基づいて曲がった線を描きます。ジオメトリタイプが曲線をサポートしていない場合、連続した小さなセグメントが曲率を近似するために使用されます。
-  ストリーム・デジタイジング: フリーハンドで線を描きます。つまり、マップキャンバスのカーソル移動と **ストリーム許容値** に従ってノードを追加します。ストリーム許容値は、連続する頂点の間隔を定義します。現在、サポートされている単位はピクセル (px) のみです。このモードでは、開始時の左クリックと終了時の右クリックのみが必要です。
-  シェープをデジタイズ: **シェープデジタイジングツールバー** のツールをトリガーして、正多角形のポリゴンを描画することができます。

デジタイジングツールを切り替えても、選択した技法は残ります。同じジオメトリを描くときに、最初の 3 つの方法のいずれかを組み合わせることができます。

地物の追加

レイヤのタイプに応じて、ツールバー上の  レコードを追加 、  点地物を追加する 、  線の地物を追加 、  ポリゴン地物を追加 アイコンを使用すると、現在のレイヤに新しい地物を追加できます。

ジオメトリなしの地物を追加するには、  レコードを追加 ボタンをクリックし、開いた地物フォームで属性値を入力します。

空間的なツールで地物を作成するには、最初にジオメトリをデジタイズし、それから属性値を入力します。ジオメトリをデジタイズするには、以下の手順で操作します：

1. (デフォルトなのでオプション)  セグメントでデジタイズ ジオメトリ描画方法を選択します
2. マップエリア上で左クリックし、新しい地物の最初の点を作成します。ポイント地物の場合はこれでおしまいで、必要に応じて属性値を入力するための地物フォームが開きます。
3. ラインやポリゴンジオメトリの場合は、左クリックでキャプチャしたい点の追加を続けます。 **地物へのスナップ** オプションや **グリッドへのスナップ** 、 **高度なデジタイズ** パネルに頼ることで、各頂点を正確な位置に落とすことができます。

ラインやポリゴンは、ひとつずつクリックしたノードを直線のセグメントで結ぶほか:

- **自動トレース** は、デジタイズを加速します。配置した頂点の間に、既存の地物に沿った連続した直線を作成します。
- フリーハンドデジタイズは、 R を押すか、 ストリーム・デジタイジング を起動します。
- 曲線として描画するには、 Ctrl+Shift+G を押すか、 曲線でデジタイズ を起動します。

注釈: ラインやポリゴンのジオメトリをデジタイズしながら、ジオメトリの描画方法を切り替えられるので、直線的なセグメント、フリーハンドのもの、曲線的なパーツを混ぜた地物を作成できます。

---

4. Delete キーまたは Backspace キーを押すと、間違って追加した最後のノードを元に戻すことができます。
5. 頂点の追加が終わったら、マップエリア上の任意の場所を右クリックして、その地物のジオメトリ入力が完了したことを確定させます。

Tip: デジタイズのラバーバンドをカスタマイズする

ポリゴンのキャプチャ中、下にある地物や点をキャプチャしたい場所をデフォルトの赤色のラバーバンドが隠してしまうことがあります。この問題は、 **設定** **オプション** **デジタイズ** メニューでラバーバンドの **塗りつぶし色** を低い不透明度（アルファチャンネル）に設定することで解決できます。 **頂点の編集中にラバーバンドを更新しない** にチェックを入れると、ラバーバンドを使用しないこともできます。

---

6. ライン地物については、 Shift キーを押しながら右クリックすることで、ラインの輪を自動的に閉じます。
7. 属性ウィンドウが表示され、新しい地物のための情報を入力します。 図 16.89 では、架空の新しい川に属性を設定しています。ところで、 **設定** **オプション** メニュー下の :guilabel:`デジタイズ` メニューでは以下の設定もできます：

- **地物作成後に属性フォームをポップアップさせない** にチェックを入れると、フォームが開くのを抑制します。
- **最後に入力した値を利用する** にチェックを入れると、フォームを開いたときに最後の入力値がフィールドに自動的に入っており、変化させる値を入力するだけでよくなります。

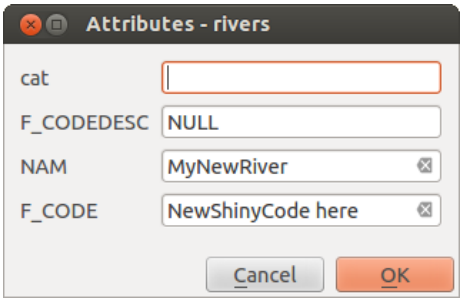


図 16.89: 新しいベクタ地物をデジタイズした後の属性値入力ダイアログ

頂点ツール

QGIS には、ベクタ地物の頂点を操作するためのツールが 2 つあります：

-  頂点ツール（現在のレイヤ）：（ **レイヤ** パネルの）アクティブなレイヤの地物のみが影響を受けます
-  頂点ツール（全レイヤ）：編集可能なすべてのレイヤの地物が影響を受けます。アクティブレイヤを切り替えることなく地物を編集したり、複数のレイヤ（例えば国とその行政区の境界）を同時に編集することができます。

編集可能なベクタレイヤに対して、頂点ツールは地物の頂点を操作する CAD プログラムと似たような機能を提供します。一度に複数の頂点を選択し、移動、追加、削除を行うことができます。また、頂点ツールはトポロジ編集機能をサポートしています。これらのツールは選択状態を維持し続けるため、何らかの操作を行った場合でも、その地物とツールの選択状態が維持されます。

設定 🔧 **オプション** **デジタイズ** **頂点編集用検索半径**: 

プロパティをゼロより大きい値に設定することが重要です。 そうでない場合、QGIS はどの頂点が編集されているか伝えることができなくなり、警告が表示されます。

Tip: 頂点マーカー

QGIS の現在のバージョンでは、さまざまな種類の頂点マーカー：「半透明円」、「クロス」、「なし」をサポートしています。 **設定** メニューから 🔧 **オプション** を選択し、 **デジタイズ** タブをクリックして該当するエントリを選択します。

基本操作

あるレイヤが編集モードにあるとして、まず最初に頂点ツールを有効にします。すると、頂点にマウスカーソルを重ねた際に赤い丸が表示されます。

- **頂点の選択** ：以下の操作で頂点を選択できます
  - `Shift` キーを押しながら頂点をクリックすることで、一度に複数選択する
  - クリック＆ドラッグで長方形を描き、ターゲットの頂点群を囲んで選択する
  - ターゲットの頂点群を囲むポリゴンを描いて選択する： `Alt` キーを押しながらクリックすることで、頂点ツールでポリゴンのデジタイズを開始します。続けてクリックするたびに、ラバーバンドポリゴンに新しい頂点を追加します。 `Backspace` キーまたは `Delete` キーを押すと、最後に追加したラバーバンドの頂点を削除します。 `Esc` キーはポリゴン選択モードをキャンセルします。これは、Backspace や Delete キーでラバーバンドのすべての頂点を削除した場合も同様です。右クリックするとポリゴンのデジタイズを終了し、ラバーバンドポリゴン内の頂点がすべて選択されます。

  頂点を選択すると、選択された頂点の色が青に変わります。現在の選択に頂点を追加するには、`Shift` キーを押しながら上記の操作を行います。現在の選択から頂点を削除するには、 `Ctrl` キーを押しながら操作します。

  Tip: 地物選択によって頂点ツールを制限する

頂点は複数の地物（やレイヤ）にまたがって選択できます。頂点が込み入った場所で特定の地物の頂点を探したい場合には、最初にその地物を選択してください。それから頂点ツールで長方形を描画したりポリゴン選択を行えば、選択した地物の頂点のみが選択されます。

これは、 **頂点エディタ** パネル内に地物を表示している場合も同様です。

- **頂点一括選択モード** ：一括選択モードは、 Shift+R キーを押すことで有効化できます。1 つ目の頂点をシングルクリックで選択し、別の頂点を **クリックせず** カーソルを重ねます。すると、（ポリゴンの場合には）最短のパスを使用して、その間にあるすべての頂点が動的に選択されます。

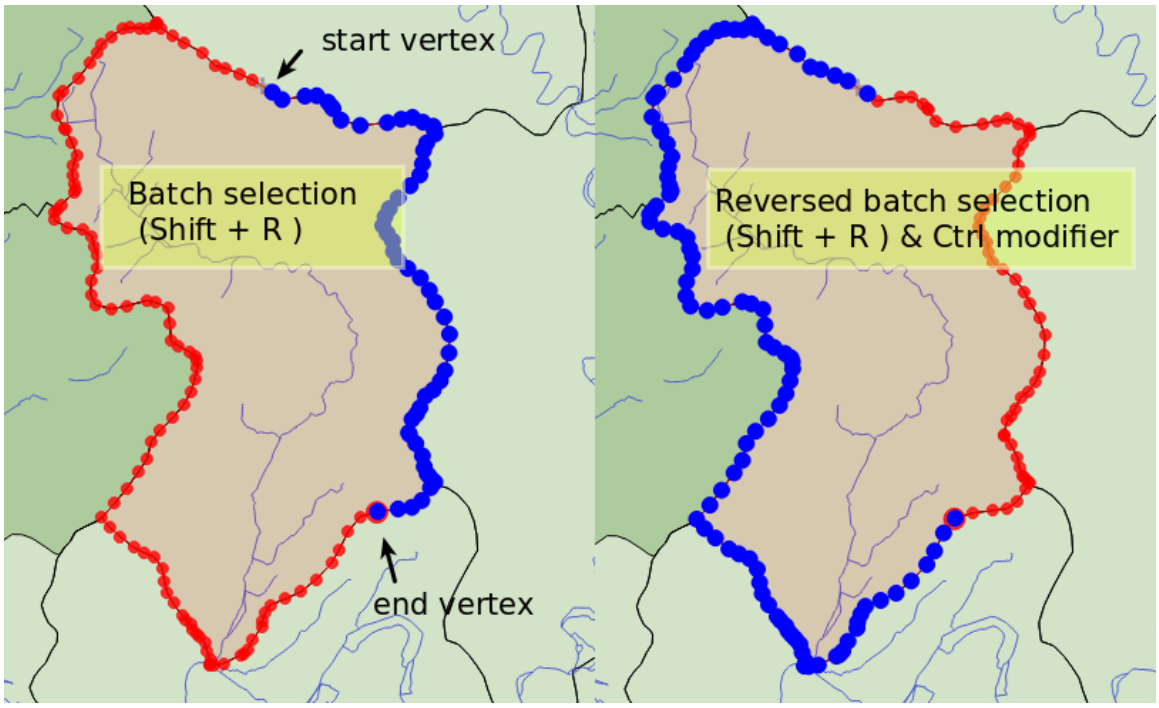


図 16.90: Shift+R を使用した頂点の一括選択

Ctrl を押すと選択を反転させ、地物境界の最長パスに沿って選択します。2 回目のクリックで頂点の選択を終了するか、 Esc キーを押して一括選択モードを抜けます。

- **頂点の追加** ：ラインジオメトリやポリゴンジオメトリに頂点を追加するには、 Shift キーを押しながらセグメント上でダブルクリックします。

  セグメントにマウスカーソルを重ね得ると、セグメントの中央に仮想の新しいノードが現れます。このノードをクリックして、カーソルを別の位置に持っていき再度クリックすると、新しい頂点が追加されます。ラインについては、 両端にも仮想ノードが用意されています。これをクリックし、さらにクリックを続け、右クリックで終了すると、既存のラインを簡単に延長することができます。

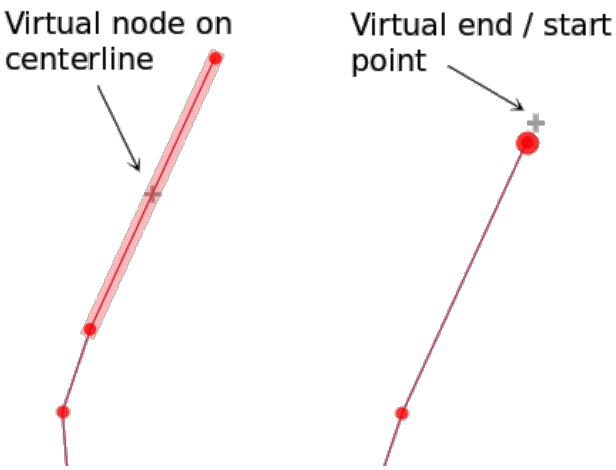


図 16.91: 頂点追加時の仮想ノード

- **頂点の削除** ：頂点を選択し、 Delete キーを押します。地物の全ての頂点を削除すると、データソースが対応しているならば、ジオメトリのない地物が生成されます。頂点の削除は地物を完全に削除するのではなく、ジオメトリ部分のみ削除することに注意してください。地物を完全に削除するには、 選択物の削除 ツールを使用します。

- **頂点の移動** ：移動させたい頂点をすべて選択し、選択した頂点または辺をクリックして、移動させたい新しい場所でもう一度クリックします。 **地物へのスナップ機能** を使用したり、 **高度なデジタイズパネル** を使用して 2 回目のクリックの前に距離や角度、正確な XY 座標位置に関して制限をかけることができます。選択したすべての頂点が平行移動します。

  ただし、 **グリッドへのスナップ** オプションが有効になっている場合には、選択した頂点は移動先の位置から最も近いグリッド交点にスナップします。選択しなかった頂点も同様に、最も近いグリッド交点へと移動します。この場合は単純な平行移動にはなりません。

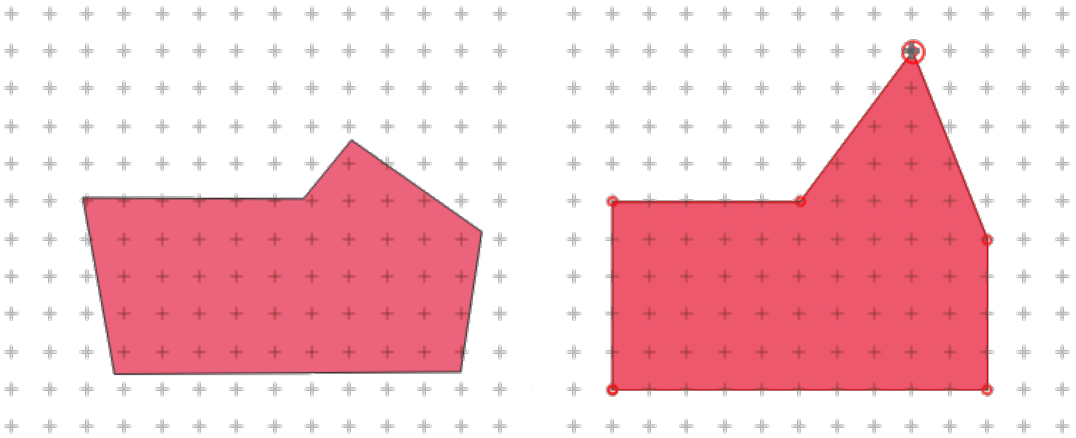


図 16.92: 上の頂点を移動させると、全ての頂点がグリッドにスナップする

- **隣接するセグメントを曲線に/曲線から変換する： 変換したいセグメントの中央の頂点を選択し、0 文字キーを押します。頂点が曲線の中にあった場合、曲線は直線に変換されます。頂点が 2 つの直線の間にあった場合、それらは曲線に変換されます。直線の最初または最後の頂点は、曲線中央の頂点には変換できません。レイヤは曲線ジオメトリタイプに対応している必要があります。

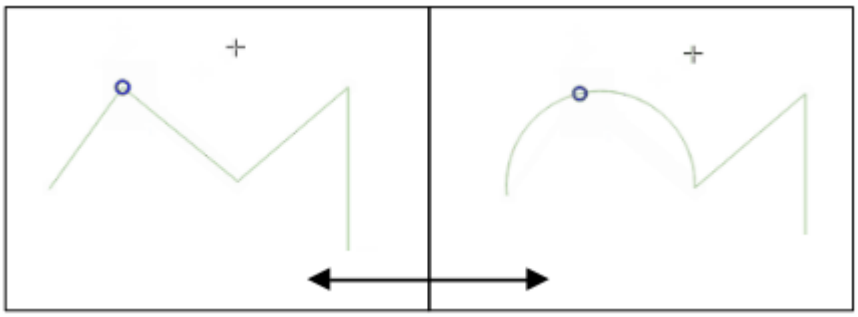


図 16.93: 0 文字で曲線から直線へ切り替える

頂点ツールで行われた各変更は、**元に戻す** ダイアログ内で個別のエントリとして格納されます。トポロジ編集が有効となっているときには、上記すべての操作がトポロジ編集をサポートしていることを覚えておいてください。また、オンザフライの投影もサポートしています。

頂点エディタパネル

頂点ツールを有効にすると、**頂点エディタ** パネルも表示されます。地物の上で右クリックすると、その地物のすべての頂点と x, y（該当する場合は z, m ）座標、 r （円形の場合は半径）のリストがパネルに表示されます。また、この地物は編集専用になり、他の地物の編集はできなくなります：

- テーブルの行を選択すると、マップキャンバスの対応する頂点が選択され、その逆もまた然りです。
- マップキャンバス上でクリックまたはドラッグすると、その地物の頂点とセグメントのみが選択または移動されます
- テーブルの座標を変更すると、頂点の位置が更新されます。頂点の Z 座標や M 値を編集するのに便利な方法です。
- また複数の行を選択し、まとめて削除することもできます。
- 新しい頂点は、囲んだ地物にのみ追加できます

頂点ツールを操作するたびに **頂点エディタ** パネルをすぐに表示させたくない場合（他のパネルを隠したり、パネルの配置を邪魔する可能性がある）、パネル上部の ☰ オプション メニューで *Auto-open table* エントリーのチェックを外してください。その後、パネルを閉じることもできます。パネルを再び開くには、パネルやツールバーの上で右クリックしてリストから選択するか、**デジタイジングツールバー** の **頂点エディタを表示** エントリにチェックを入れる必要があるでしょう。

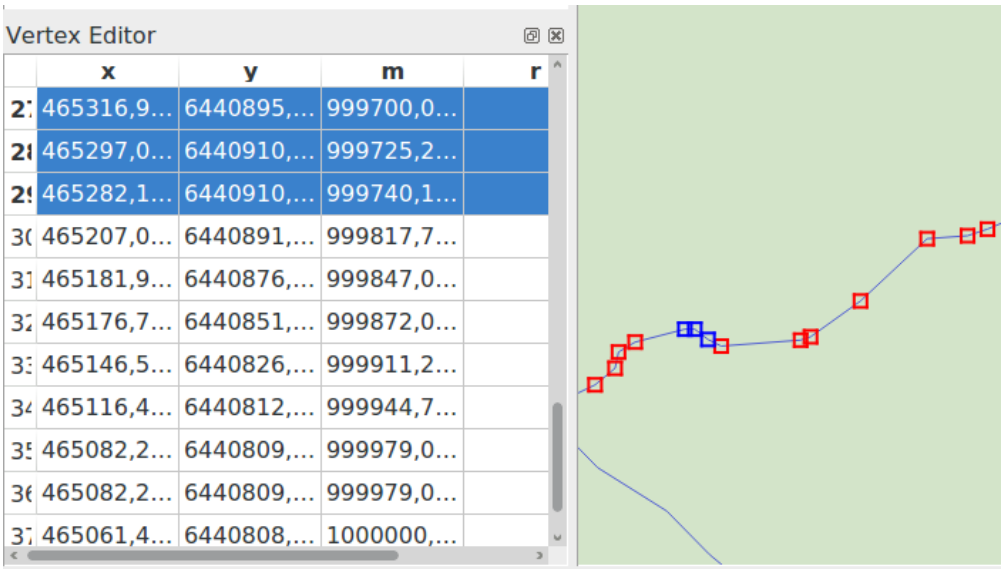


図 16.94: 選択したノードを示す頂点エディタパネル

Z 座標または M 値の割り当てルール

3D ベクタ地物や M 値を持つ地物のデジタイズは、(X,Y) 2D レイヤのものとさほど変わりありません。この章で説明したツールやオプションはここでも利用可能で、頂点やポイントを平面的な環境に配置するのに役立ちます。その後、Z 座標（または M 値）の割り当てを処理する必要がある場合があります：

- デフォルトでは、QGIS は新しい頂点に **設定　オプション　デジタイズ** タブで設定された **デフォルトの *Z* 値**（それぞれ **デフォルトの *M* 値**）を割り当てます。**高度なデジタイズパネル** が使用されている場合は、その z（それぞれ m）ウィジェットから値が取得されます。
- 頂点にスナップする場合、新しい頂点または移動した頂点はスナップした頂点の Z または M の値を取ります。
- トポロジー編集がオンの状態でセグメントにスナップすると、新しい頂点の Z 値または M 値がセグメントに沿って補間されます。
- **高度なデジタイズパネル** の z（それぞれ m）ウィジェットが  ロックされている場合、その値が頂点に適用され、スナップした頂点やセグメントの Z や M の値よりも優先されます。

既存の地物の Z 値や M 値を編集するには、**頂点編集パネル** を使うことができます。カスタム Z 値やカスタム M 値を持つ地物を作成するには、**高度なデジタイズパネル** を利用するとよいかもしれません。

地物の切り取り、コピーと貼り付け

選択した地物は、同じ QGIS プロジェクト内のレイヤ間で切り取り、コピー、貼り付けすることができます。ただし、貼り付け先のレイヤはあらかじめ  編集モード切替 で編集モードに設定されている必要があります。

Tip: コピー＆ペーストを使用してポリゴンをラインに変換、ラインをポリゴンに変換

ライン地物をコピーしてポリゴンレイヤに貼り付けると、QGIS はライン地物の両端を結んで閉じたジオメトリに対応する境界を持つポリゴンをターゲットレイヤに貼り付けます。これは、同じデータから異なるジオメトリを生成するための簡単な方法です。

地物は外部アプリケーションにテキストとして貼り付けることもできます。つまり、地物は CSV 形式で表現され、ジオメトリデータは OGC Well-Known Text（WKT）形式で表示されます。QGIS 外の WKT 地物および GeoJSON 地物は QGIS 内のレイヤに貼り付けることもできます。

コピー＆ペースト機能はどんなときに便利でしょうか？これまでの説明で、一度に複数のレイヤを編集したり、レイヤ間で地物をコピー＆ペーストできることはわかりました。なぜこのようなことが必要なのでしょうか？例えば、新しいレイヤでとある作業をする必要がありますが、必要なのは 1 つか 2 つの湖だけだとしましょう。 `big_lakes` レイヤにある 5,000 個の湖は必要ありません。このようなときは新規レイヤを作成し、コピー＆ペーストを使って必要な湖をポトンと貼り付けることができます。

例として、新しいレイヤに湖をいくつかコピーしてみましょう：

1. コピーしたいレイヤを読み込みます（ソースレイヤ）
2. 貼り付けたいレイヤを読み込みまたは作成します（ターゲットレイヤ）
3. ターゲットレイヤを編集モードにします
4. 凡例内でレイヤをクリックして、ソースレイヤをアクティブにします
5.  シングルクリックによる地物選択 ツールを使用して、ソースレイヤ上の地物（複数可）を選択します
6.  地物のコピー ツールをクリックします
7. 凡例の貼り付け先レイヤをクリックしてアクティブにします
8.  地物の貼り付け ツールをクリックします
9. 編集モードを終了して変更内容を保存して下さい

ソースレイヤとターゲットレイヤでスキーマが異なる（フィールド名やデータ型が異なる）場合にはどうなるでしょうか？ QGIS は、一致するものは入力し、一致しない残りは無視します。ターゲットレイヤにコピーされる属性を気にしないのであれば、フィールド名やデータ型の設計は重要ではありません。地物とその属性をすべて確実にコピーしたい場合には、スキーマを一致させるように注意してください。

注釈: 貼り付け地物の一致

ソースレイヤとターゲットレイヤが同じ投影法を使用している場合、貼り付けられた地物はソースレイヤと同じジオメトリになります。しかし、ターゲットレイヤの投影法が異なる場合には、QGIS はジオメト

リが同一であることを保証できません。これは単に、投影法間の変換時に小さな丸め誤差が入り込むためです。

Tip: 文字列属性を他のレイヤにコピーする

属性テーブルに「文字列」型の新しいカラムを作成し、それより長さの大きい属性カラムから値を貼り付ける場合、そのカラムサイズの長さは同じ長さに拡張されます。これは、GDAL Shapefile ドライバが、挿入されるデータの長さに動的に対応するために、文字列と整数フィールドを自動拡張することを知っているためです。

選択地物の削除

地物全体（属性とジオメトリ）を削除するには、まずは通常の  シングルクリックによる地物選択 ツールを使用してジオメトリを選択します。また、選択は属性テーブルから行うこともできます。選択が完了したら、`Delete` または `Backspace` キーを押すか、  選択物の削除 ツールを使用して、地物の削除ができます。選択された複数の地物を一度に削除することができます。

デジタイジングツールバーの  地物の切り取り ツールでも地物の削除ができます。このツールは地物を事実上削除しますが、これを「空間クリップボード」にも置きます。このため、地物の切り取りを削除する目的でも使うことができます。切り取った地物はその後、  地物の貼り付け ツールを使用して戻すことができるので、1 段階のみの取り消し機能とも言えます。切り取り、コピー、貼り付けは現在の地物選択に対して機能するので、一度に複数の地物の操作ができます。

元に戻すとやり直し

 元に戻す と  やり直す ツールを使用すると、ベクタレイヤの編集操作の取り消しややり直しができます。元に戻す/やり直す履歴（ 図 16.95 参照）にある全ての操作を表示する、ドッキング可能なウィジェットもあります。このウィジェットはデフォルトでは表示されていません。ツールバーを右クリックして **元に戻す/やり直す パネル** のチェックボックスを有効にすると、このウィジェットを表示することができます。ただし、たとえウィジェットが表示されていなくとも、この「元に戻す/やり直す」機能は有効です。

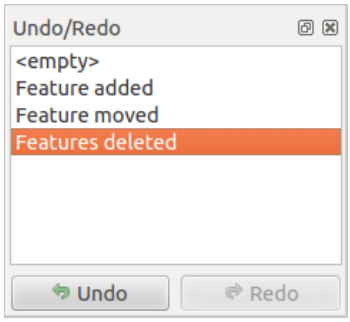


図 16.95: デジタイズ操作を元に戻す・やり直す

「元に戻す」を押すか Ctrl+Z （または Cmd+Z ）を押すと、すべての地物および属性の状態が、取り消す操作が起こった前の状態に戻ります。通常のベクタの編集操作以外の変更（例えば、プラグインによって行われた変更）については、どのような変更が行われたかによって、元に戻せる場合と戻せない場合があります。

元に戻す/やり直し 履歴ウィジェットを使用するには、履歴リストから操作を単純にクリックするだけです。すべて地物は、選択した操作を行った後の状態に戻されます。

レイヤ編集内容の保存

レイヤが編集モードになっているとき、変更はすべて QGIS のメモリ内にとどまっています。従って、変更はすぐにデータソースやディスクにコミット/保存されるわけではありません。現在のレイヤの編集内容を保存したいが、編集モードは離れることなく編集を続けたい場合には、  レイヤ編集内容の保存 ボタンをクリックします。  編集モード切替 で編集モードをオフにする（または QGIS を終了する）と、変更を保存するか破棄するかを尋ねられます。

変更が保存できない場合（例えばディスクが一杯であったり、属性が範囲外の値の場合など）、QGIS のメモリ内の状態はそのままです。これにより、編集を訂正して保存をやり直すことができます。

Tip: データの整合性

編集を開始する前には常に、データソースをバックアップすることをお勧めします。QGIS 製作者はデータの整合性を維持するためにあらゆる努力をしていますが、この点に関する保証はありません。

一度に複数のレイヤを保存する

レイヤの編集機能は、複数のレイヤのデジタイズができます。複数のレイヤで行ったすべての変更を保存するには、  **選択レイヤの保存** を選択します。また、選択した全てのレイヤのデジタイズ内容を取りやめるために、  **選択レイヤをロールバック** を行うこともできます。選択したレイヤの編集を終了したい場合には、  **選択レイヤの編集キャンセル** が簡単な方法です。

プロジェクトの編集中の全レイヤを対象とした同じ機能もあります。

Tip: トランザクショングループを使用して、複数のレイヤの変更を一度に編集・保存・ロールバックする

同じ PostGreSQL データベースの複数レイヤで作業する場合、**プロジェクト　プロパティ...　データソース** の **可能な場合は自動的にトランザクショングループを作成する** オプションを有効にすると、動作を同期させる（編集モードのオンオフや変更の保存・ロールバックを同時に行う）ことができます。

16.3.5 高度なデジタイズ

表 16.4: ベクタレイヤの高度な編集ツールバー

アイコン	目的	アイコン	目的
	高度なデジタイズツールの有効化		
	地物の移動		地物をコピー／移動
	地物を回転		地物を簡素化
	地物をスケーリング		
	リングを追加		部分を追加
	リングを充填		線の向きを反転
	リングを削除		部分を削除
	曲線をオフセット		地物の変形
	部分の分割		地物を分割
	選択地物の属性結合		選択地物を結合
	点のシンボルを回転		点のシンボルのオフセット
	地物をトリム/延長		

地物の移動


地物の移動 ツールを使用して、既存の地物の移動ができます：

1. 移動させたい地物を選択します。
2. マップキャンバス上をクリックして、変位の原点を指定します。スナップ機能によって正確な点を選択することもできます。

   高度なデジタイズによる制限 を活用して、原点座標を正確に設定することもできます。この場合には、次のようにします：

   1. 最初に、  ボタンを押してパネルを有効化します。
   2. **x** キーを押し、使用したい原点に対応する x 座標値を入力します。その後、このオプションの横にある  ボタンを押して、値をロックします。
   3. **y** 座標についても同様に操作します。
   4. マップキャンバスをクリックすると、原点が指定した座標に配置されます。
3. マップキャンバス上を移動して、ポイントの移動先を指定します。ここでも、移動の終点を配置するためにスナップモードを使用したり、上記のように高度なデジタイズパネルを使用して相補的な 距離 と 角度 による配置制約を指定したりすることができます。
4. マップキャンバス上をクリックすると、地物全体が新しい位置に移動します。

同様にして、 地物のコピーと移動 ツールを使用して、地物の移動したコピーを生成することもできます。

注釈: **地物の移動** や **地物のコピーと移動** ツールを使ってマップキャンバス上を最初にクリックしたときに地物が何も選択されていない場合、マウスの下にある地物だけがこの操作の影響を受けます。そのため、複数の地物を移動させたい場合には、それらの地物を先に選択しておく必要があります。

地物を回転

地物の回転 ツールを使用すると、マップキャンバス内で 1 つまたは複数の地物を回転させます。

1. 地物の回転 アイコンをクリックします。
2. 回転させたい地物をクリックします。地物の重心が回転の中心として参照され、回転後の地物のプレビューが表示されます。また、ウィジェットが開き、現在の *Rotation* 角度が表示されます。
3. 満足する新しい配置角度でマップキャンバス上をクリックするか、テキストボックスに回転角を入力します。 **スナップ先 °** ボックスを使用して、回転の値に制限をつけることもできます。
4. 複数の地物を同時に回転させたい場合には、最初にそれらを選択しておきます。デフォルトでは、結合したジオメトリの重心の周りに回転します。

デフォルトの地物重心とは異なるアンカーポイントを使用することもできます。 `Ctrl` キーを押しながらマップキャンバスをクリックすると、その点が新しい回転の中心となります。

地図上をクリックする前に `Shift` キーを押したままの場合、回転は 45 度ずつで行われます。これはユーザー入力ウィジェットの中で後から変更できます。

地物の回転を中止するには、 `ESC` キーを押すか 地物の回転 アイコンをクリックします。

地物をスケーリング

地物をスケーリング ツールは、地物の回転と似たようなツールです。選択した地物の回転を行う代わりに、ジオメトリの再スケーリングを行います。変更はアンカーポイントを基準にして行われ、拡大縮小の比率はキャンバス上部の角に表示されるウィジェットで手動で指定することもできます。

地物を簡素化

地物の簡素化 ツールを使用すると、ジオメトリの有効性を保つ範囲で頂点の数を減らしたりスムージングすることで、ラインやポリゴンのジオメトリをインタラクティブに変形させることができます。

1. 地物の簡素化 ツールを選択します。
2. 地物をクリックするか、地物の上で矩形をドラッグします。

3. ダイアログがポップアップし、適用する **方法** を指定できます。つまり、実行したい事柄に応じて以下のとおり指定します：

   - **ジオメトリの簡素化** 、つまりオリジナルよりも頂点の数を減らしたい場合には、利用可能な手法は 距離で簡略化する 、 グリッドで簡素化 、 面積で簡略化（Visvalingam）です。簡略化に使用する **許容範囲** の値を レイヤの単位 、 ピクセル または 地図単位 で指定する必要があります。許容範囲の値を大きくするほど、より多くの頂点が削除されます。
   - 新しい頂点を追加して **ジオメトリのスムージング** を行いたい場合には、スムージング オプションを指定します。既存の各頂点に対して、2 つの頂点がその頂点から伸びるセグメント上に配置されます。 **オフセット** は、頂点が配置される距離のセグメントの長さに対する割合を表します。また、この配置処理の **反復** 数を設定することができます。反復数が大きいほど、より多数の頂点が配置され、よりスムーズな地物になります。

   使用した設定はプロジェクトの終了時や編集セッションの終了時に保存されます。このため、次回、地物の簡素化を行う場合に同じパラメータで行うことができます。

4. ダイアログの下部には適用される修正の概要が表示され、地物数と頂点数（操作前の数と操作後の数および変更の比率）が一覧表示されます。また、マップキャンバスでは、適用後のジオメトリが既存のジオメトリ上にラバーバンド色で表示されます。

5. 適用後のジオメトリが期待通りのものならば、 *OK* ボタンを押して修正を適用します。そうでない場合には、 **キャンセル** ボタンを押すかマップキャンバスで右クリックし、操作を中断します。

注釈: レンダリングのためだけにジオメトリを簡素化する **設定** ▸ **オプション** ▸ **レンダリング** メニュー内の地物の簡素化オプションとは異なり、 地物の簡素化 ツールは、永続的にデータソース内の地物ジオメトリを変更します。

部分を追加

選択された地物に 部分を追加 して、マルチポイント、マルチラインまたはマルチポリゴン地物を生成します。新しく生成する部分は、既存の地物の外側にデジタイズしなければなりません。部分の追加は、既存の地物をあらかじめ選択してから実行する必要があります。

部分を追加 は、ジオメトリのない地物にジオメトリを追加するためにも使用できます。まず、属性テーブルで地物を選択し、 部分を追加 ツールで新しいジオメトリをデジタイズします。

部分を削除

部分の削除 ツールは、マルチパート地物から部分を削除できます（例えば、マルチポリゴン地物からポリゴンを削除する）。このツールは、すべてのマルチパートジオメトリ（ポイント、ライン、ポリゴン）で動作します。このツールはさらに、地物のジオメトリ要素を完全に除去するためにも使用できます。部分を削除するには、単に対象の部分の内部をクリックします。

リングを追加

ツールバーの  リングを追加 アイコンを使用して、リングポリゴンを作成できます。これは、既存のポリゴン領域の内部にさらにポリゴンをデジタイズできて、それが「穴」となり、外側と内側のポリゴンの境界の間の領域のみがリングポリゴンとして残ります。

リングを充填

 リングの充填 ツールを使うと、他のポリゴン地物の完全に内部にある、重複する領域が何もないポリゴン地物を作成できます。つまり、この新しい地物は既存の地物の穴を覆います。このような地物を作成するには、以下の手順で操作します：

1.  リングの重点 ツールを選択します。
2. 既存の地物上に新しいポリゴンを描きます。QGIS はそのジオメトリに（  リングを追加 ツールを使ったときのように）リングを追加し、ジオメトリがリングに一致する新しい地物（  ポリゴン地物を追加 ツールで内部境界に沿って **トレース** したような地物）を作成します。
3. また、地物にすでにリングがある場合には、リング上にマウスを乗せて `Shift` キーを押しながら左クリックすると、その場所に穴を充填する新しい地物が作成されます。

   新しく作成した地物の **地物属性** フォームが開きます。フォームには「親」の地物の値や **フィールドの制約** による値があらかじめ入力されています。

リングを削除

 リングの削除 ツールは、穴の内側をクリックして既存のポリゴン内部のリングを削除できます。このツールは、ポリゴンとマルチポリゴンの地物に対応しています。このツールをポリゴンのリングの外側で使用したときには、ポリゴンは何も変わりません。

地物の変形

ツールバーの  地物の変形 ツールを使用すると、ラインやポリゴン地物の形状を変形できます。ラインの場合には、元のラインとの最初の交点から最後の交点までの部分を置き換えます。

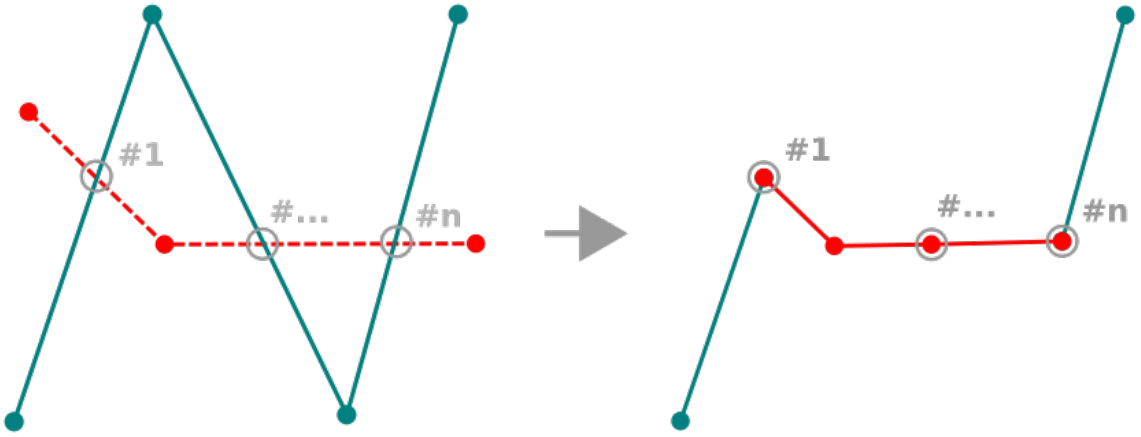


図 16.96: ラインの変形

Tip: 変形ツールでラインストリングジオメトリを延長する

地物の変形 ツールを使用して、既存のラインストリングジオメトリの延長ができます。ラインの最初または最後の頂点にスナップして、新しいラインを描画します。ジオメトリの有効性を確認した後、地物は 2 つのラインを結合したジオメトリになります。

ポリゴンの場合には、ポリゴンの境界を変形します。これが機能するためには、変形ツールの線はポリゴンの境界を少なくとも 2 回横切る必要があります。ラインを描画するには、マップキャンバスをクリックして頂点を追加します。終了するには右クリックしてください。ラインと同様に、最初の交点と最後の交点の間のセグメントのみが考慮されます。ポリゴンの内側にある変形ラインのセグメントはポリゴンを切り取り、ポリゴンの外側にある変形ラインのセグメントはポリゴンを拡大させます。

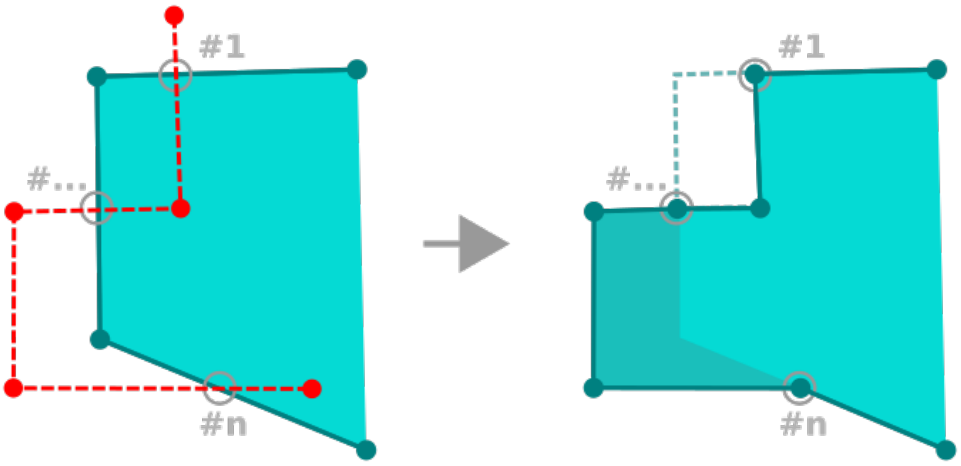


図 16.97: ポリゴンの変形

ポリゴンでは、変形によって意図しない結果が生じることがあります。地物の変形ツールは主にポリゴンの小さな部分を置き換えるのに便利なツールであって、大規模な形状の見直しには向いていません。また、

無効なポリゴンを生成してしまうため、変形ラインは複数のポリゴンリングを横切ることはできません。

注釈: 変形ツールは、ポリゴンのリングや閉じた線の開始位置を変更することがあります。つまり「二回」表される開始位置のポイントは、変形後はもはや同じではありません。これは、ほとんどのアプリケーションでは問題とならないかもしれませんが、留意すべき点です。

曲線のオフセット


曲線のオフセット ツールは、ラインレイヤが平行移動したものを作成します。このツールは編集中のレイヤに適用することもできますし（ジオメトリが変更されます）、背景レイヤに適用することもできます（この場合はラインやリングのコピーを作成して編集中のレイヤに追加します）。このため、このツールは距離線のレイヤの作成に最適です。 **ユーザー入力** ダイアログがポップアップし、変位距離が表示されます。

平行移動したラインレイヤを作成するには、まずは編集モードに入り、 
曲線のオフセット ツールをアクティブにします。次に、平行移動させたい地物をクリックします。マウスを移動して好きな場所をクリックするか、ユーザー入力ウィジェットに平行移動させたい距離を入力します。2 回目のクリックの際に `Ctrl` キーを押しながらクリックすると、オフセットのコピーを作成します。変更は 
レイヤ編集内容の保存 ツールで保存できます。

QGIS のオプションダイアログ（デジタイズタブの **曲線オフセットツール** セクション）では、 **継ぎ目スタイル** 、 **象限セグメント** 、 **miter 制限** といったパラメータを設定できます。

線の向きの反転

ラインジオメトリの向きを変える機能は、地図作成やネットワーク解析の準備のために役立つことがあります。

ラインの方向を変えるには：

1. 
線の向きの反転 をクリックして、線の向きの反転ツールをアクティブにします。
2. ライン上をクリックすると、線の向きが反転します。

地物を分割


地物を分割 ツールを使用して、地物を 2 つまたはそれ以上の独立した地物に分割することができます。独立した地物の各ジオメトリが属性テーブル内の新しい行に対応します。

ライン地物またはポリゴン地物を分割するには：

1. 
地物を分割 ツールを選択します。
2. 分割したい地物を横切るように線を引きます。選択がアクティブな場合には、選択された地物のみが分割されます。分割が設定できたら、対応するフィールドには **デフォルトの値または式** が適用され、その他の親地物の属性はデフォルトで新しい地物にコピーされます。

3. その後は、分割された地物の属性を通常通りに変更することができます。

Tip: 1 クリックでポリラインを新しい地物に分割する


切断箇所 ツールを使用して、ポリライン地物の既存の頂点をスナップしクリックすると、その地物が 2 つの新しい地物に分割されます。

部分の分割

QGIS では、部分の個数が増えるようにマルチパート地物の部分を分割することができます。 
部分の分割 アイコンを使用して、分割したい部分を横切るように線を引くだけです。

Tip: 1 クリックでポリラインを新しい部分に分割する


部分の分割 ツールを使用して、ポリライン地物の既存の頂点をスナップしクリックすると、その地物が同じ地物に属する 2 つの新しいポリライン部分に分割されます。

選択地物の結合


選択地物の結合 ツールを使用すると、既存の地物をマージして新しい地物を作成できます。ジオメトリを結合して、新しいジオメトリを生成します。地物に共通の境界がない場合には、マルチポリゴン/マルチライン/マルチポイント地物が作成されます。

1. 最初に、結合させたい地物群を選択します。
2. 次に、 
選択地物の結合 ボタンを押します。
3. 新しく開いたダイアログで、表の最下段にある **結合** 行には、結合結果の地物の属性値が表示されています。この属性値は以下の方法で変更することが可能です：
   - 対応するセルにマニュアル入力値を入力する
   - テーブルの行を選択し、 **選択した地物から属性を取得する** ボタンを押して、この元の地物の値を使用する
   - **もっとも長い地物の属性のみ** を押して、最も長いライン地物、最も大きなポリゴン、最も多くのパートを持つマルチポイントの属性値を利用する
   - **すべてのフィールドをスキップ** ボタンを押して、属性値を空にする
   - テーブルの上部にあるドロップダウンメニューを展開して、対応するフィールドにのみ適用する上記の任意のオプションを選択する。ここでは、元の地物の属性の集約値（フィールドの型に応じて、最小値、最大、中央値、合計、カウント、文字列の連結などがある。関数の完全なリストは **統計量の出力パネル** を参照）を選択することもできる

注釈: レイヤがフィールドにデフォルト値やデフォルト式を持つ場合には、それが結合した地物の初期値として使用されます。

4. *OK* ボタンを押して、修正を適用します。単一の地物（またはマルチパート地物）がレイヤに作成され、選択していた地物と置き換わります。

選択地物の属性結合


選択地物の属性結合 ツールを使用すると、地物の境界をマージすることなく、同じ属性を地物に適用できます。ダイアログは「選択地物の結合」ツールと同じですが、選択されたオブジェクトの属性の一部は同じになる一方で、ジオメトリはそのままである点が異なります。

点のシンボルを回転


点のシンボルの回転 は、マップキャンバスでポイントシンボルの回転を個別に変更することができます。

1. 最初に、回転の値を保存するフィールドを指定する必要があります。これは、以下の手順でシンボルの **データによって定義された** 回転のプロパティにフィールドを割り当てることで行います：

   1. **レイヤプロパティ　　シンボロジ** ダイアログで、シンボルの編集ダイアログを見つけます。
   2. シンボルレイヤの（なるべく）最上位の **マーカー** レベルの **回転** オプションの近くにある、  **データによって定義された上書き** ウィジェットをクリックします。
   3. **フィールドの型** コンボボックスでフィールドを選択します。これにより、このフィールドの値は、これに応じて各地物のシンボルを回転させるために使われます。

      または、 
データをプロジェクトに格納する エントリをチェックして、回転値を制御するための **補助テーブル** フィールドを生成することもできます。

注釈: すべてのシンボルレイヤに同じフィールドを割り当てるよう注意してください

データ定義の回転フィールドをシンボルツリーの最上位に設定すると、自動的にすべてのシンボルレイヤに設定が伝搬します。すべてのシンボルレイヤに同じ設定がされていることが、 **点のシンボルの回転** ツールでグラフィカルなシンボル回転を行うための前提条件です。実際、シンボルレイヤの一つが回転プロパティに異なるフィールドを持っている場合には、ツールは機能しません。

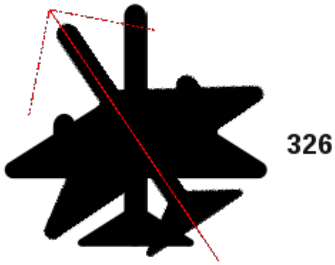


図 16.98: 点のシンボルの回転

2. 次に、点のシンボルの回転 ツールを使用して、マップキャンバス内でポイントシンボルをクリックします。
3. 周りでマウスを動かします。回転値付きの赤い矢印が表示されます（ 図 16.98 参照）。 `Ctrl` キーを押しながらマウスを動かすと、回転は 15 度単位で行われます。
4. 求める角度の値となったら、もう一度クリックします。シンボルがこの新しい回転角でレンダリングされ、対応するフィールドがこれに応じて更新されます。

   右クリックすると、シンボルの回転を中止します。

点のシンボルのオフセット

点のシンボルのオフセット を使うと、マップキャンバス内でポイントシンボルがレンダリングされる位置をインタラクティブに変更することができます。このツールは This tool behaves like the 点のシンボルの回転 ツールと同様に動作しますが、シンボルの各レイヤでデータ定義の **オフセット（X,Y）** プロパティにフィールドを結合させる必要がある点が異なります。このフィールドには、シンボルがマップキャンバスで動かされた場合に地物のオフセット量が入力されます。

1. シンボルの **オフセット（X,Y）** プロパティのデータ定義ウィジェットにフィールドを関連付けます。シンボルが多数のレイヤからなる場合には、各レイヤにフィールドを割り当てられます。
2. 点のシンボルのオフセット ツールを選択します。
3. ポイントシンボルをクリックします。
4. マウスを新しい位置に移動します。
5. もう一度クリックします。シンボルが新しい位置に移動します。元の位置からのオフセット値がリンクしたフィールドに格納されます。

   右クリックすると、シンボルのオフセットを中止します。

注釈: 点のシンボルのオフセット ツールは、点地物自体は移動させません。点地物自体を移動させる目的では、頂点ツール（現在のレイヤ） または 地物の移動 ツールを使う必要があります。

地物をトリム/延長

 トリム/延長 ツールを使用すると、（マルチ）ラインや（マルチ）ポリゴンジオメトリのセグメントを選択したセグメント（切断線）に一致するように短縮または延長できます。この結果、変更されたジオメトリはターゲットのセグメントまたはその延長線上にスナップする頂点を持ちます。選択されたジオメトリが互いにどのように関連して配置されているかによって、このツールは以下のいずれかを行います：

- **トリム** ：ラインのセグメントやポリゴンの境界のうち、切断線を越える部分を削除する
- **延長** ：ポリゴンの境界やラインのセグメントを切断線にスナップするように延長する

既存のジオメトリをトリムまたは延長するには：

1. 関係するレイヤ（群）のセグメントに対して、適切に **スナップ設定** を有効化します
2.  トリム/延長 ツールを選択します
3. 対象とする限界セグメント、つまりは他のセグメントの延長またはトリムに関して限界となるセグメントをクリックします。このセグメントはハイライト表示されます。
4. トリムまたは延長させたいセグメントにカーソルを移動させます。これはジオメトリの最後のセグメントである必要はありませんが、アクティブレイヤ上にある必要があります。
5. トリムまたは延長させたいセグメント上にカーソルを乗せると、QGIS は地物のジオメトリがどのようになるかプレビューを表示します。OK ならば、セグメントをクリックしてください。トリムの場合には、短くする方の部分を選択する必要があります。
6. 両方のセグメントが 3D の場合には、ツールは Z 値を得るために制限セグメントに対して内挿を実行します。

注意:  トリム/延長 ツールを使用する際には、変更されたジオメトリに注意してください。入力によっては、このツールは不正なジオメトリを作成することがあり、レイヤの保存時に失敗する可能性があります。

16.3.6 シェープデジタイジング

シェープデジタイジング ツールバーは、  シェープをデジタイズ *geometry drawing method* が **高度なデジタイズツールバー** で選択したものと同期しています。これは、規則的な形状のライン又はポリゴン地物を描くための一連のツールを提供します。

Circular string by radius

 Circular string by radius ボタンは、曲線上の 2 つのノードと半径を指定して、円形のジオメトリを持つライン又はポリゴン地物を追加することができます：

1. 左クリックを 2 回すると、ジオメトリ上に 2 点が配置されます。
2. マップキャンバスの右上にある **半径** ウィジェットは、現在の半径（ポイント間の距離に相当）を表示します。そのフィールドを好きな値に編集してください。
3. カーソルを移動すると、これらの制約に一致する円弧の概要が表示されます。期待した円弧が表示されたら、右クリックして検証します。
4. 新しい点を追加して、別の円弧のシェーピングを開始します。

注釈: 曲線ジオメトリが曲線ジオメトリとして保存されるのは対応しているデータプロバイダのみ

QGIS では、任意の編集可能なデータ形式で曲線ジオメトリのデジタイズを行うことができますが、地物を曲線ジオメトリとして保存するためには、曲線ジオメトリをサポートするデータプロバイダ（PostGIS、メモリレイヤ、GML、WFS など）を使用する必要があります。サポートしないプロバイダに保存すると、QGIS は円弧をセグメント化します。

円を描く

円を描くためのツールセットがあります。各ツールの説明は以下の通りです。

円は円形ストリングに変換されます。従って、 *Circular string by radius* で説明したように、データプロバイダが対応しているならばこれは曲線ジオメトリとして保存され、対応していないならば QGIS は円弧をセグメント化します。

-  2 点から円： 2 点は円の直径と向きを定義します。（左クリック、右クリック）
-  3 点から円： 円上の既知の 3 点から円を描画します。（左クリック、左クリック、右クリック）
-  中心点と別の点で円： 指定された中心と円上の点を持つ円を描画します（左クリック、右クリック）。**高度なデジタイズパネル** と共に使用すると、最初のクリックの後に距離の値を設定しロックすることで、このツールは「中心と半径から円を追加する」ツールになることができます。
-  3 本の接線から円： 3 つのセグメントの接線上にある円を描きます。**セグメントにスナップを必ず有効にすることに注意**（**スナップ許容範囲と検索半径の設定** を参照。） セグメントをクリックすると、接線を追加することができます。2 つの接線が平行である場合、最初の平行な接線をクリックした座標が円の位置を決定するのに使われます。3 本の接線が平行な場合は、エラーメッセージが表示され、入力はクリアされます。（左クリック、左クリック、右クリック）
-  2 本の接線と点から円： 3 本の接線から円と似ていますが、2 つの接線を選択し、半径を入力し、希望の中心を選択する必要がある点が異なります。

楕円を描く

楕円を描くためのツールセットがあります。各ツールの説明は以下の通りです。

楕円は円形ストリングに変換することができないため、これは常にセグメント化されます。

-  中心と 2 点から楕円： 指定された中心、長軸、短軸を持つ楕円を描画します。（左クリック、左クリック、右クリック）
-  中心と点から楕円： 楕円を中心と角のあるバウンディングボックスに描画します。（左クリック、右クリック）
-  領域範囲から楕円： 楕円を対向する 2 つの角を持つバウンディングボックスに描画します。（左クリック、右クリック）
-  フォーカスから楕円： 焦点となる 2 点と楕円上の 1 点により楕円を描画します。（左クリック、左クリック、右クリック）

長方形を描く

長方形を描くためのツールセットがあります。各ツールの説明は以下の通りです。

-  中心と点で長方形を追加 ：中心と角の 1 点で長方形を描きます。（左クリック、右クリック）
-  領域範囲の長方形を追加 ：2 つの対角で長方形を描きます。（左クリック、右クリック）
-  3 点で長方形を追加（第 2 点と第 3 点からの距離）：3 つの点から方向のある長方形を描きます。最初の点と第 2 点は最初の辺の長さと角度を決定します。第 3 点がもう 1 つの辺の長さを決定します。辺の長さを設定するために、 **高度なデジタイズパネル** を使用することができます。（左クリック、左クリック、右クリック）
-  3 点で長方形を追加（点 p1、p2 のセグメント上に投影された点からの距離）：上のツールと同様ですが、2 つ目の辺の長さは第 3 点を最初の辺上に投影した点から計算されます。（左クリック、左クリック、右クリック）

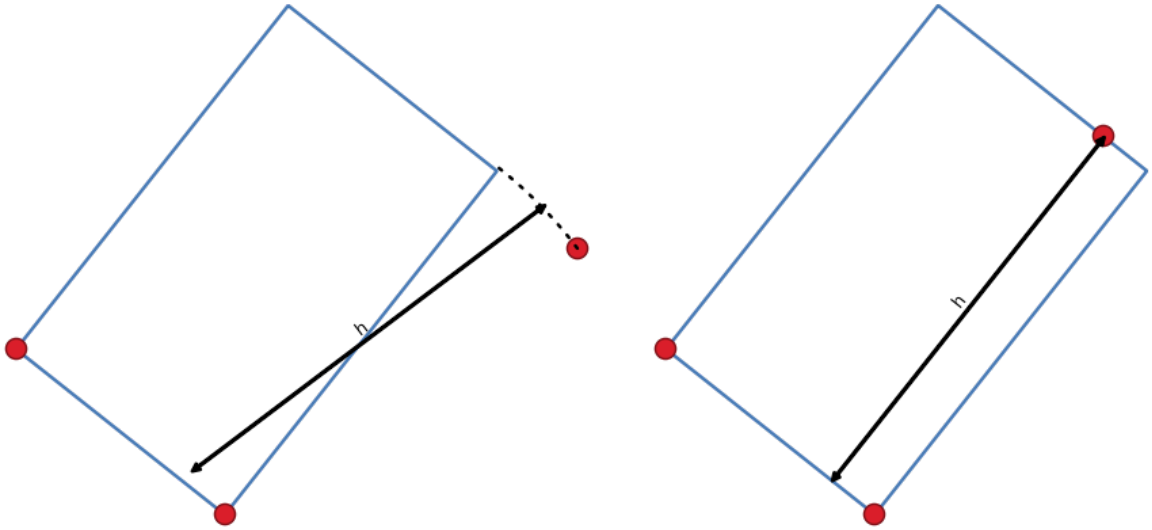


図 16.99: 3 点から距離（右）と投影（左）を使用して長方形を描く

正多角形を描く

正多角形を描くためのツールセットがあります。各ツールの説明は以下の通りです。左クリックして最初の点を配置するとダイアログが現れ、多角形の辺の数を設定できます。右クリックで正多角形の描画を終了します。

- 2 点で正多角形を追加 ：2 点で最初の辺の長さと角度を決定して正多角形を描きます。
- 中心と点で正多角形を追加 ：与えられた中心点からの正多角形を描きます。第 2 点は辺の 1 つの角度と中点への距離を決定します。
- 中心と角点で正多角形を追加 ：上のツールと同様ですが、第 2 点は頂点への角度と距離を決定します。

16.3.7 高度なデジタイズパネル

新規または既存のジオメトリをキャプチャ、リシェイプ、分割する際、高度なデジタイズパネルを使用することもできます。特定の角度に平行または垂直な線を正確にデジタイズしたり、特定の角度に線をロックすることができます。さらに、X、Y 座標、3D 地物の場合は Z 座標、M 値を入力することで、新しいジオメトリを正確に定義することができます。

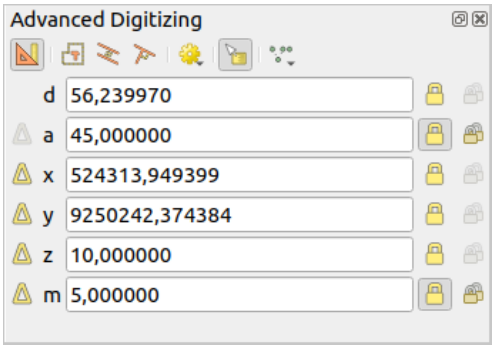


図 16.100: 高度なデジタイズパネル

高度なデジタイズ パネルは、ツールバー上で右クリックするか、**ビュー** **パネル** メニューから、または Ctrl+4 を押すことで開くことができます。パネルが表示されたら、高度なデジタイズツールの有効化 ボタンをクリックして、ツールのセットを有効にします。

注釈: このツールは、マップビューが地理的座標系の場合には有効になりません。

高度なデジタイズツールの目的は、マップキャンバスでデジタイズする最中、マウスを動かすときの座標、長さ、および角度をロックすることです。

相対参照または絶対参照を使用して、制約を作成することもできます。相対参照とは、次の頂点制約の値が前の頂点またはセグメントを基準とすることを意味します。

ツールバー

高度なデジタイズパネル の上部には、以下のボタンがあります：

- 高度なデジタイズツールの有効化
- 作図モード を使用すると、クリックした位置をキャプチャして、距離や角度、X、Y、Z、M 値の相対値を固定するための参照点として利用できます。詳細は **作図モード** を参照してください。
- 平行 既存の線に平行な線を引くために使用します（詳細は **平行線と垂直線** 参照）
- 垂直 既存の線と垂直な線を引くために使用します（詳細は **平行線と垂直線** 参照）
- 共通角にスナップする ：カーソルを動かすとスナップできる仮想の線が表示され、次の頂点を追加するために利用できます。スナップ可能な線は、最後に追加した頂点とプリセットのリスト（5 °、10 °、15 °、18 °、22.5 °、30 °、45 °、90 ° 刻み）から選択した角度（絶対角度または前のセグメントに対する相対角度）によって定義されます。この機能を無効にするには、 **共通角にスナップしない** を選択します。
- Floater を切り替え: カーソルのすぐ横に座標のライブプレビューを表示し、素早くデジタル化することができます。値は **パネルのショートカット** を使ってアクセスすることができ、編集したり、検証（ Enter を押す）後に ロック したりすることができます。

- :sup:`作図ツール `は、既存の要素の外挿座標に基づいて頂点の配置を制約するいくつかのオプションを提供します：
  - *Line Extension*: セグメントにカーソルを合わせると、マップキャンバスにセグメントを延長する紫の点線が表示されます。この仮想線上の任意の場所に頂点をスナップすることができます。
  - *X/Y Point*: 頂点にカーソルを合わせると、そのXまたはY座標に沿って、マップキャンバス上に紫色の点線が表示されます。この仮想線上の任意の場所に頂点をスナップすることができます。2つの異なる頂点にカーソルを合わせると、両方の仮想座標線が生成され、その交点にスナップすることも可能です。

ツールバーの下には、デフォルトでマップキャンバス内のカーソルの位置や移動が反映されたテキストボックスが多数用意されています。これらの値を編集することで、編集するアイテムの位置を固定することができます：

- *d* 基準となる位置（通常は最後に編集した頂点）からの距離
- *a* 基準となる位置（通常は最後に編集したセグメント）からの角度（絶対または相対）
- *x* ポインタのX座標
- *y* ポインタのY座標
- *z* でデフォルトのZ値、またはポインタの下にある頂点またはセグメントのZ座標を指定します。
- *m* はデフォルトのM値、またはポインタの下にある頂点やセグメントのM値

キーボードショートカット

高度なデジタイズパネルを使った作業をスピードアップするためのキーボードショートカットがいくつかあります：

表 16.5: 高度なデジタイズパネルツールのキーボードショートカット

キー	キー単独	Ctrl+ または Alt+	Shift+
D	距離を設定	距離をロック	
A	角度を設定	角度をロック	最後のセグメントに対する相対的な角度に切り替え
X	X座標を設定	X座標をロック	最後の頂点に対する相対的なXに切り替え
Y	Y座標を設定	Y座標をロック	最後の頂点に対する相対的なYに切り替え
Z	Z座標を設定	Z座標をロック	最後の頂点に対する相対的なZに切り替え
M	M値を設定	M値をロック	最後の頂点に対する相対的なM値に切り替え
C	作図モードを切り替え		
P	垂直モードと平行モードを切り替え		

注釈: Z座標とM値のオプションは、レイヤのジオメトリ次元が対応している場合にのみ利用可能です。

絶対参照デジタイジング

新しいジオメトリを最初から描画するときには、指定した座標から頂点のデジタイズを開始できると非常に便利です。

例えば、ポリゴンレイヤに新しい地物を追加するために  ボタンをクリックしたとします。地物の編集を開始したい正確な座標を次のようにして入力できます：

1. *x* テキストボックスをクリックします（またはキーボードショートカット **X** を使います）。
2. 開始したい X 座標の値を入力し、**Enter** を押すか、右にある  ボタンをクリックして、マップキャンバス上でマウスの X 軸をロックします。
3. *y* テキストボックスをクリックします（またはキーボードショートカット **Y** を使います）。
4. 開始したい Y 座標の値を入力し、**Enter** を押すか、右にある  ボタンをクリックして、マップキャンバス上でマウスの Y 軸をロックします。
5. レイヤーに Z 座標または M 値がある場合、対応する *z* または *m* ウィジェットが有効になり、 **設定 オプション デジタイズ** タブで設定したデフォルト値を表示します。
   1. *z* または *m* テキストボックスをクリックします（または **Z** や **M** キーボードショートカットを使います）。
   2. 欲しい座標値を入力して **Enter** を押すか、その右にある  ボタンをクリックしてウィジェットに値をロックします。

注釈: 既存の地物から Z 座標と M 値を自動的に決定する方法については、 ***Z* 座標または *M* 値の割り当てルール** を参照してください。

---

6. 2 本の青い点線と緑の十字は、入力された正確な座標を識別します。マップキャンバス上でクリックすると、緑の十字の位置に頂点が追加されます。

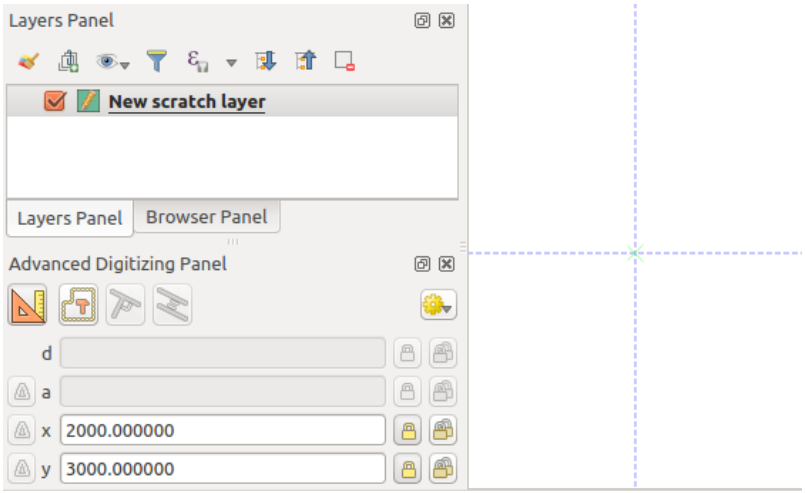


図 16.101: 指定の座標から描画を開始

7. 上記のように、次の頂点のために新しい座標セットを追加して進むこともできますし、別の *mode of digitizing*（セグメント、曲線、ストリームなど）に切り替えることもできます。

8. 指定した長さのセグメントを描画したい場合には：

   1. *d* **（距離）** テキストボックスをクリックします（またはキーボードショートカット D ）

   2. （マップの単位で）距離の値を入力します。

   3. Enter キーを押すか、右にある  ボタンをクリックして、マップキャンバスのマウス位置をこのセグメント長さになるようにロックします。マップキャンバスでは、距離テキストボックスに入力した値を半径とする円によって最後の頂点が囲まれます。円周上のバツ印が、クリックした場合の次の頂点の位置を表します。

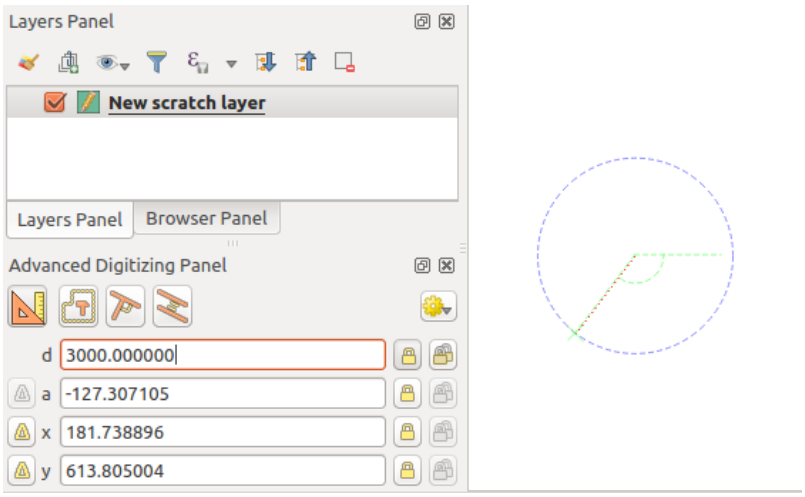


図 16.102: 固定長セグメント

9. また、セグメントの角度を設定することで頂点位置を制限することもできます。前述と同様に、以下のように操作します：

   1. *a* **（角度）** テキストボックスをクリックします（またはキーボードショートカット A ）

   2. （度単位で）角度の値を入力します。

   3. Enter キーを押すか、右にある  ボタンをクリックして、角度をロックします。最後の頂点を通り、設定した角度で回転した線がマップキャンバスに表示されます。線上のバツ印が、クリックした場合の次の頂点の位置を表します。

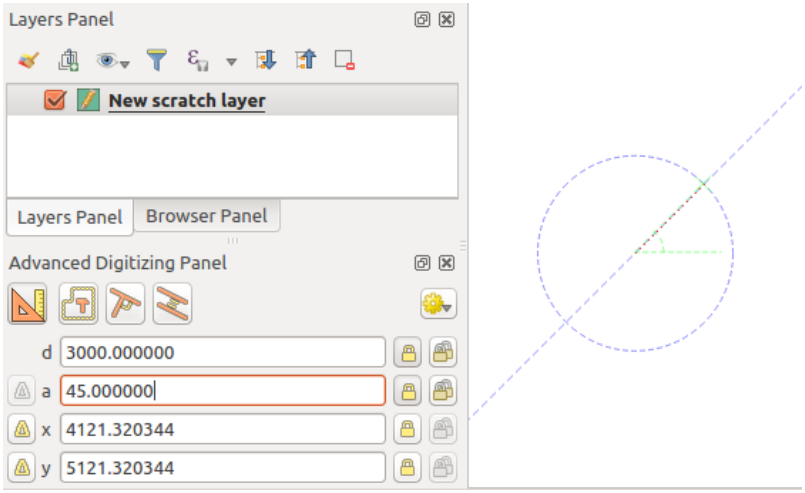


図 16.103: 固定角セグメント

ヒント: Ctrl+<key> または Alt+<key> を押すと、ターゲットとするプロパティを自動的にロックし、値の編集状態になります。値を編集して Enter キーを押せば完了です。 Floater の切り替え と一緒に使用すれば、キーボードでデジタイズができ、非常に時間短縮になります。

相対参照デジタイジング

角度または座標の絶対値を使用する代わりに、最後にデジタイズされた頂点またはセグメントを基準にした角度や座標を使用することもできます。

角度については、*a* テキストボックスの左にある ボタンをクリックする（または Shift+A を押す）と、ひとつ前のセグメントに対する相対角度に切り替わります。このオプションをオンにすると、最後のセグメントとマウスポインターの間の角度が計測されます。

座標については、*x*、*y*、*z*、*m* テキストボックスの左にある ボタンをクリックする（または Shift+<key> を押す）と、ひとつ前の頂点に対する相対的な座標に切り替わります。このオプションをオンにすると、最後の頂点を座標設定の原点とみなして座標計測します。

継続的な固定

絶対参照のデジタイズ、相対参照のデジタイズのどちらでも、 **常に固定する** ボタンをクリックすることで、角度や距離、X、Y、Z、M 値の制約を継続的に固定することができます。常に固定を使用すると、複数の点や頂点を同じ制約条件でデジタイズすることができます。

平行線と垂直線

上記のツールはすべて、 垂直 および 平行 ツールと組み合わせることができます。これらの 2 つのツールは、別のセグメントに対して完全に垂直または平行なセグメントを描画できます。ターゲットとなるセグメントは他のレイヤのセグメントでも良いですし、同じレイヤ内の別の地物や、デジタイズ中の地物のセグメント（**自己スナップオプション** の設定が必要）とすることもできます。

垂直 なセグメントを描画するには：

1. セグメントの頂点の一つを追加します。
2. 垂直 アイコンをクリックして（またはキーボードショートカット P ）垂直モードを有効にします。
3. 描画するセグメントに対して垂直としたいセグメントをクリックします。
4. 最初の頂点を通り、クリックしたセグメントに対して垂直な仮想点線が表示されます。角度プロパティはロックされ、次の頂点は仮想点線上に制限されます。バツ印は、カーソル位置を仮想点線上に射影した位置を表します。クリックすると、新しい頂点を配置します。

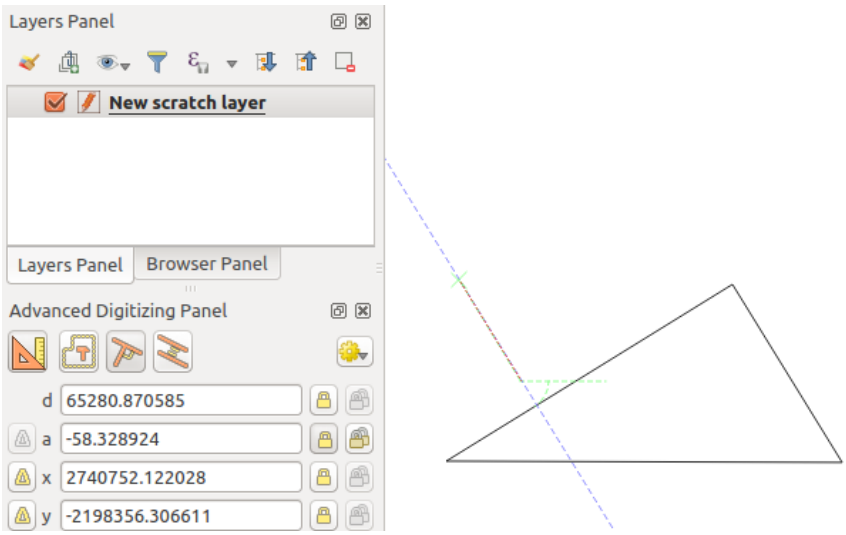


図 16.104: 垂直デジタイズ

平行 なセグメントを描画する場合も上と同様ですが、 平行 アイコンをクリックする（またはキーボードショートカット P を 2 回押す）必要がある点だけが異なります。

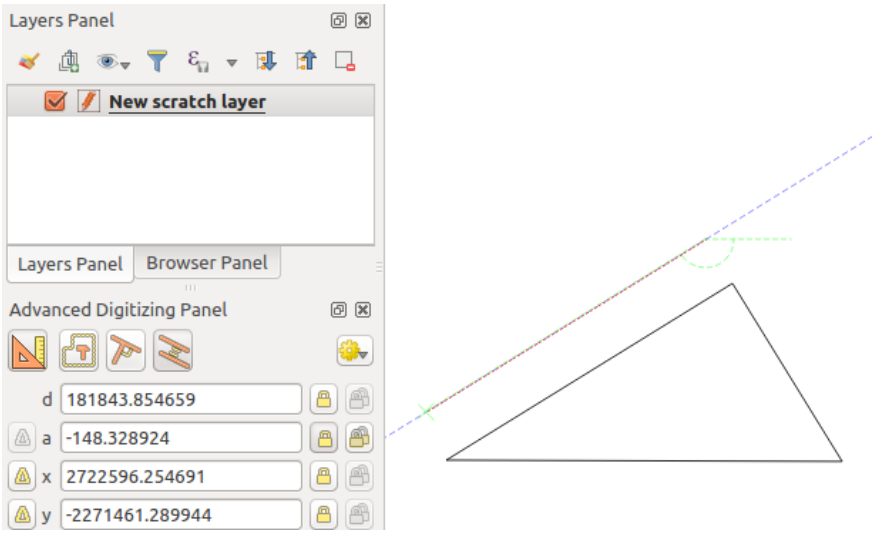


図 16.105: 平行デジタイジング

これらの 2 つのツールは単に、垂直や平行となるちょうどの角度を見つけて、編集中に角度パラメータをロックするだけです。角度パラメータのロックを解除すると、デジタイズ中のこれらの 2 つのツールの使用をキャンセルします。

作図モード

作図モード アイコンをクリックするか、キーボードショートカット C で **作図モード** 有効化・無効化を切り替えられます。作図モード中では、マップキャンバスをクリックしても新しい頂点は追加されませんが、クリックした位置はキャプチャされるので、距離や角度、X、Y、Z、M 値の相対値をロックするための基準点として使用できます。

例えば、作図モードを使用すれば、既存の点から正確にとある距離に位置する点を描画できます。

マップキャンバス内に既存の点があり、スナップモードが正しく有効になっていると、そこから所定の距離や角度で他の点を簡単に描画できます。 ボタンに加えて、作図モード アイコンをクリックするかキーボードショートカット C を使用して、 **作図モード** を有効にする必要があります。

次に、距離を計算したいポイントをクリックし、 *d* ボックスをクリックし（または D ショートカット）希望の距離を入力して、 Enter を押してマップキャンバス内でマウスの位置を固定します。

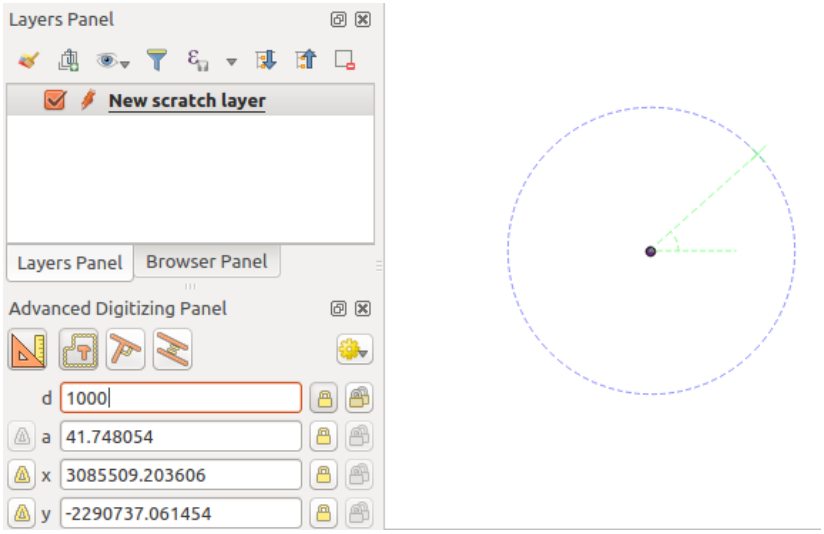


図 16.106: 点からの距離

新しい点を追加する前に C を押して作図モードを終了してください。これで地図上でクリックすると、入力された距離に点が配置されます。

また角度の制約も使用することで、例えば、元の点と同じ距離にあるが、新たに追加された点から特定の角度にある別の点を作成することもできます。 作図 アイコンをクリックするか、キーボードショートカット C で作図モードに入ります。最後に追加したポイントをクリックし、次にもう 1 点をクリックして方向セグメントを設定します。続いて、*d* テキストボックスをクリックして（またはショートカット D）、希望する距離を入力し Enter を押します。それから *a* テキストボックスをクリックし（またはショートカット A）希望する角度を入力して Enter を押します。マウスの位置は、距離と角度の両方で固定されます。

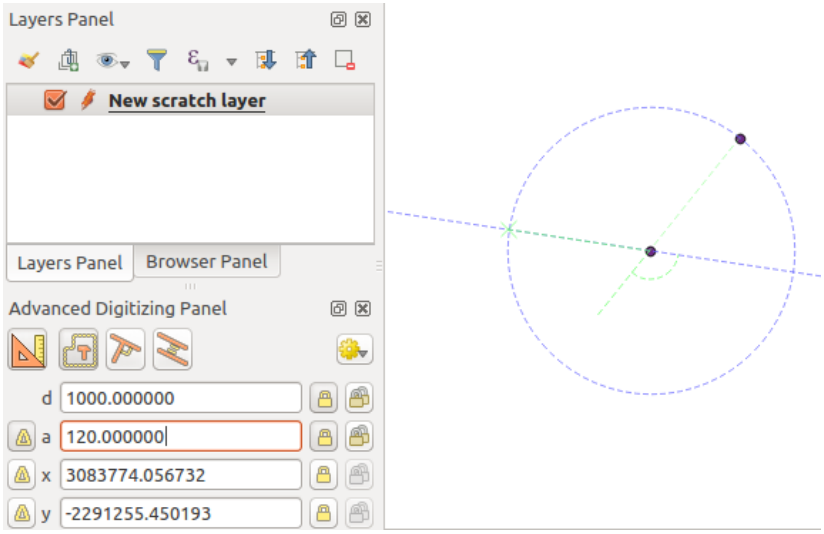


図 16.107: 点からの距離と角度

新しいポイントを追加する前に、 C を押して作図モードを終了します。これでマップキャンバスをクリックすると、入力した距離と角度でポイントが配置されます。この手順を繰り返して、複数のポイントを追加できます。

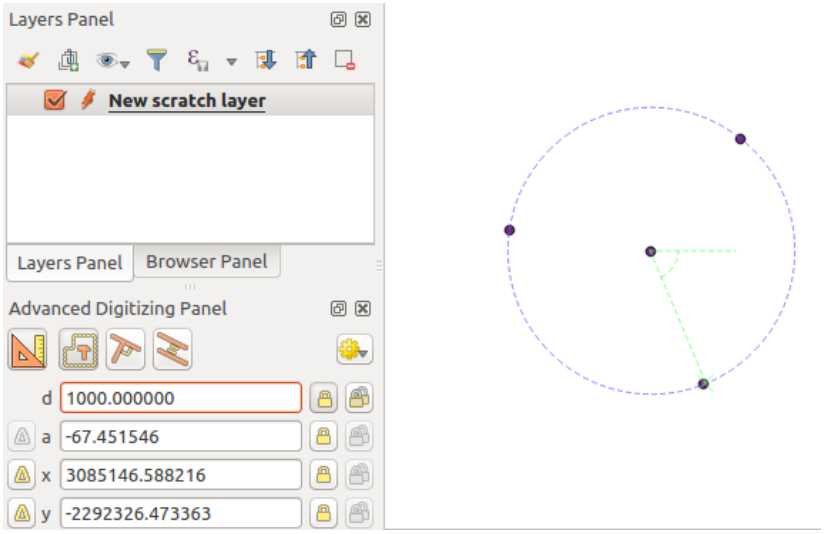


図 16.108: 所定の距離と角度の点

16.3.8 プロセシングによるレイヤのインプレース修正

プロセシングメニュー には、入力された地物のプロパティや他の地物との関係（同じレイヤ内かどうかに関わらず）に基づいて解析し、新しい地物を作成するための大規模なツールセットがあります。一般的な動作では出力として新しいレイヤを作成しますが、いくつかのアルゴリズムでは入力レイヤの修正も可能です。これは、高度で複雑な手続きを使用した複数の地物の修正を自動化するために便利な方法です。

地物をインプレースで編集するには：

1. **レイヤ** パネルで編集したいレイヤを選択します。
2. 該当する地物を選択します。このステップを省略することもできますが、その場合、修正はレイヤ全体に適用されます。
3. **プロセシングツールボックス** の上部にある In-Place 編集 ボタンを押します。アルゴリズムのリストがフィルタされ、インプレースの修正に対応したものだけが表示されます。すなわち：
   - レイヤレベルではなく、地物ソースレベルで動作するもの
   - レイヤの構造を変えないもの。例えば、フィールドの追加や削除を行わないもの
   - ジオメトリタイプを変えないもの。例えば、ラインレイヤからポイントレイヤに変換したりしないもの

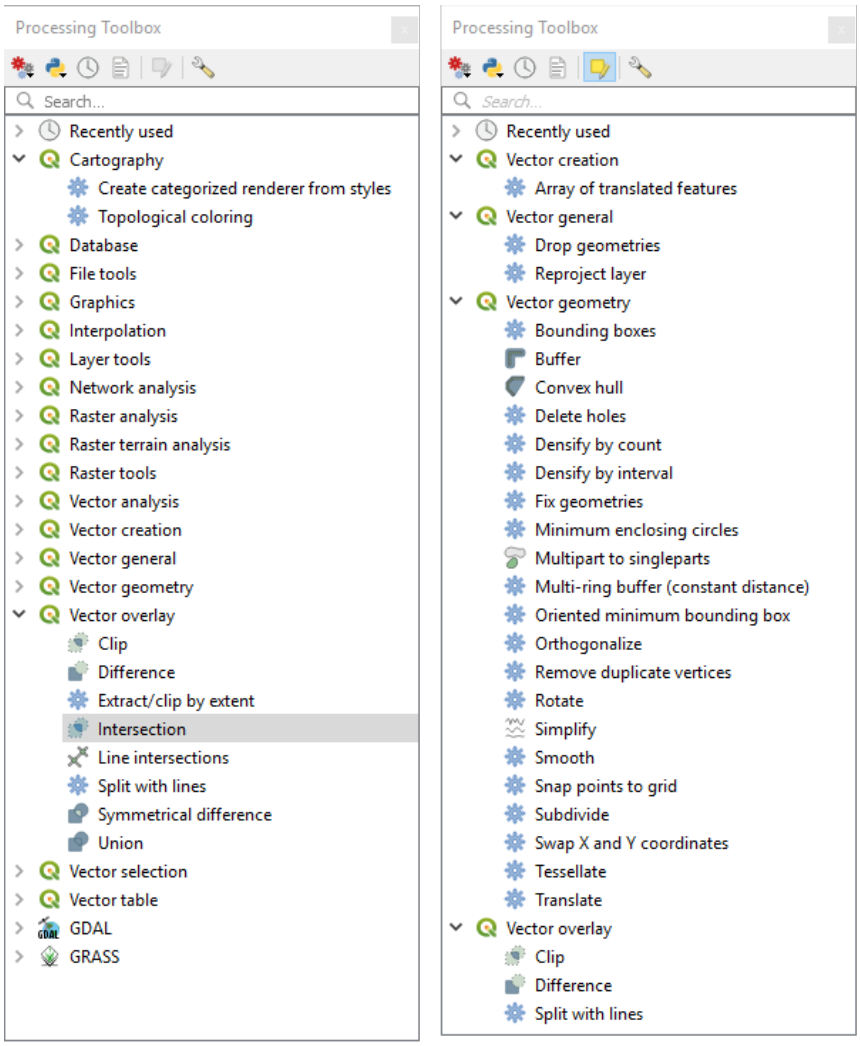


図 16.109: プロセシングアルゴリズム：全部（左） vs ポリゴンレイヤの in-place 編集対応（右）

4. 実行したいアルゴリズムを探し出し、それをダブルクリックします。

注釈: アルゴリズムが追加のユーザー設定パラメータ（通常の入出力レイヤパラメータ以外）を何も必要としない場合には、ダイアログのポップアップ無しでアルゴリズムが直ちに実行されます。

---

   1. 通常の入力レイヤと出力レイヤ以外のパラメータが必要な場合には、アルゴリズムのダイアログがポップアップします。必要な情報を入力してください。
   2. アクティブな選択があるかどうかに応じて、 **選択地物の変更** または **全地物の変更** をクリックします。

   変更がレイヤに適用され、編集履歴に置かれます。つまり、レイヤ名の横に アイコンが表示されて、レイヤは実際に編集モードに切り替わり、変更は未保存の状態となっています。

5. 通常どおり、 レイヤ編集内容の保存 を押すと、レイヤへの変更がコミットされます。また、 元に戻す を押すと変更全体をロールバックできます。

第17章　ラスタデータの操作

17.1 ラスタプロパティダイアログ

ラスタレイヤのプロパティを表示・設定するには、マップ凡例のレイヤ名をダブルクリックするか、レイヤ名を右クリックして、コンテキストメニューから **プロパティ** を選択します。これにより **ラスタレイヤプロパティ** ダイアログが表示されます。

このダイアログにはさまざまなタブがあります：

情報	ソース	シンボロジ [1]
透明度 [1]	ヒストグラム [1]	レンダリング
時系列	標高	ピラミッド
メタデータ	凡例	*QGIS* サーバー
外部プラグイン [2] タブ		

[1] **レイヤスタイルパネル** からも利用可能です

[2] インストールした **外部プラグイン** は、任意でこのダイアログにタブを追加することがあります。これらについては、このドキュメントでは説明しません。外部プラグインのドキュメントを参照してください。

Tip: ライブアップデートレンダリング

レイヤスタイル設定パネル はレイヤプロパティダイアログの一般的な機能のいくつかを提供しており、レイヤスタイルの設定を素早く行い、変更内容をマップキャンバス上で確認できる、優れたモードレスのウィジェットです。

注釈: 埋め込まれたレイヤ（ **外部プロジェクトからのレイヤの埋め込み** を参照）のプロパティ（シンボロジ、ラベル、アクション、デフォルト値、フォーム...）は、元のプロジェクトファイルから引用されているため、この動作を壊す可能性のある変更を避けるために、埋め込まれたレイヤに対してはレイヤプロパティダイアログは利用できなくなっています。

17.1.1 情報プロパティ

情報 タブは読み取り専用で、現在のレイヤの要約された情報やメタデータをさっと掴むことができる興味深い場所です。提供される情報には、以下のものがあります：

- 一般情報：プロジェクト内での名前、ソースへのパス、付随的なファイルのリスト、最終更新時刻、ファイルの大きさ、使用しているプロバイダ
- レイヤのプロバイダからの情報：領域、幅、高さ、データタイプ、GDAL ドライバ、バンドの統計情報
- 空間参照システム（CRS）：CRS の名前、単位、投影法、精度、参照（静的か動的か）
- レイヤのプロパティから読み取った情報：データタイプ、領域、幅/高さ、圧縮、ピクセルの大きさ、バンドの統計量、バンド数、行・列の数、ラスタの No-Data 値...
- **入力されたメタデータ** からの情報：アクセス、領域、リンク、連絡先、履歴など

17.1.2 ソースプロパティ

ソース タブは、選択されたラスタに関する以下のような基本的な情報を表示します：

- **レイヤパネル** で表示される **レイヤ名**
- **設定された** *CRS* ：レイヤの **座標参照系** *(CRS)* を表示します。最近使用した CRS をドロップダウンリストから選ぶか、 CRS の選択 ボタン（ **座標参照系セレクタ** を参照）をクリックすることで、レイヤの CRS を変更できます。レイヤの CRS が間違っている場合か、CRS が何も設定されていない場合にのみ、この操作を行ってください。データを再投影したい場合には、プロセシングアルゴリズムのレイヤの再投影を使用するか、 **新しいデータセットとして保存** してください。

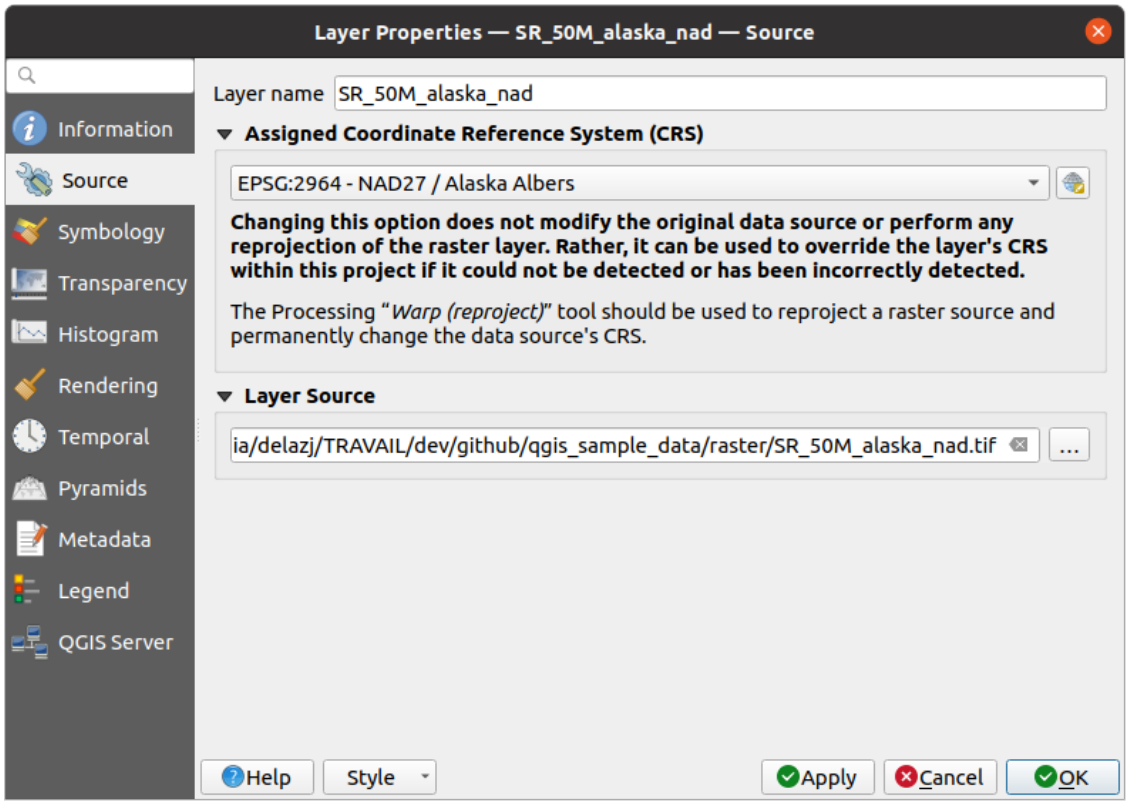


図 17.1: ラスタレイヤプロパティ - ソースダイアログ

17.1.3 シンボロジプロパティ

ラスタレイヤのシンボロジタブは、3 つの異なるセクションからなります：

- **バンドレンダリング** ：使用するレンダラーの種類を操作できます
- **レイヤレンダリング** ：レンダリングされるデータに効果を適用します
- **リサンプリング** ：マップ上でのレンダリングを最適化するリサンプリング手法を設定します

バンドレンダリング

QGIS には数多くの異なる **レンダリングタイプ** があります。データタイプと強調したい情報に応じてレンダラーを選択してください。

1. **マルチバンドカラー** - ファイルに複数のバンドがある場合（例：マルチバンドの衛星画像）
2. **カテゴリ値パレット** - インデックス付けされたパレットを持つ単バンドのファイル（例：デジタル地形図）や、ラスタレイヤをレンダリングするためのパレットの一般用途
3. **単バンドグレー** - 画像（の 1 バンド）をグレースケールでレンダリングします。ファイルがマルチバンドではなく、パレットも持たない場合には、QGIS はこのレンダラを選択します（例：陰影起伏図など）

4. **単バンド疑似カラー** - このレンダラーは連続値パレットやカラーマップのファイルに使用します（例：標高地図など）
5. **陰影図（*hillshade*）** - バンドの一つから陰影図を作成します。
6. **等高線（*Contours*）** - ソースラスタバンドに対してオンザフライで等高線を生成します。

マルチバンドカラー

マルチバンドカラーレンダラーを選択した場合には、画像から選択した 3 つのバンドがカラー画像の赤、緑、青成分として使用されます。QGIS はラスタの各バンドから自動的に **最小** と **最大** の値を取得し、色付けをこれに応じてスケーリングします。 **最小/最大値設定** のセクションで値の範囲を制御することができます。

コントラスト は次の値をとります：「強調しない」、「最小最大範囲に引き伸ばす」、「最小最大範囲に引き伸ばしカット」、「最小値から最大値までの範囲以外は無視」

注釈: コントラスト

GRASS ラスタを追加する場合、たとえ QGIS の一般オプションで他の値に設定したとしても、 **コントラスト** オプションは常に自動的に **最小最大範囲に引き伸ばす** に設定されます。

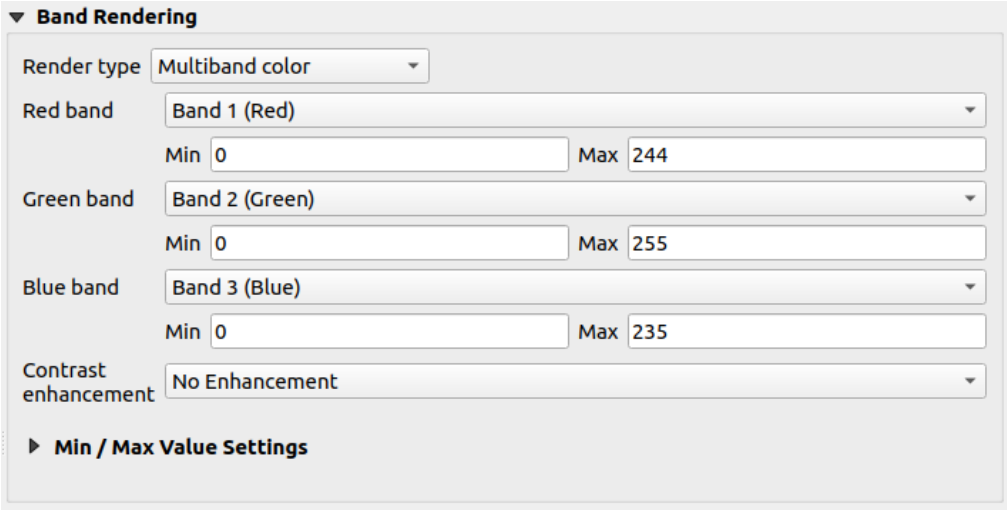


図 17.2: ラスタシンボロジ - マルチバンドカラーレンダリング

Tip: マルチバンドラスタの単バンドを表示する

マルチバンド画像の単バンド（例えば赤）を表示したい場合、緑と青のバンドを **未設定** に設定すればよいと思うかもしれません。しかし、この場合に推奨される方法は、画像のレンダリングタイプを **単バンドグレー** に設定し、使用する **グレーバンド** に赤のバンドを選択することです。

カテゴリ値パレット

これは、各ピクセル値にある特定の色が割り当てられたカラーテーブルを含んだ単バンドファイルに対する標準的なレンダラーオプションです。この場合、パレットは自動的にレンダリングされます。

これはあらゆる種類のラスタバンドに使用することができ、ユニークなラスタ値それぞれに色を割り当てます。

色を変更したい場合には、色をダブルクリックすると **色の選択** ダイアログが表示されます。

色にラベルを割り当てることもできます。このラベルはラスタレイヤの凡例で表示されます。

カラーテーブルの選択した行の上で右クリックすると、以下のコンテキストメニューが表示されます：

- 選択した行の **色を変更**...
- 選択した行の **不透明度を変更**...
- 選択した行の **ラベルを変更**...

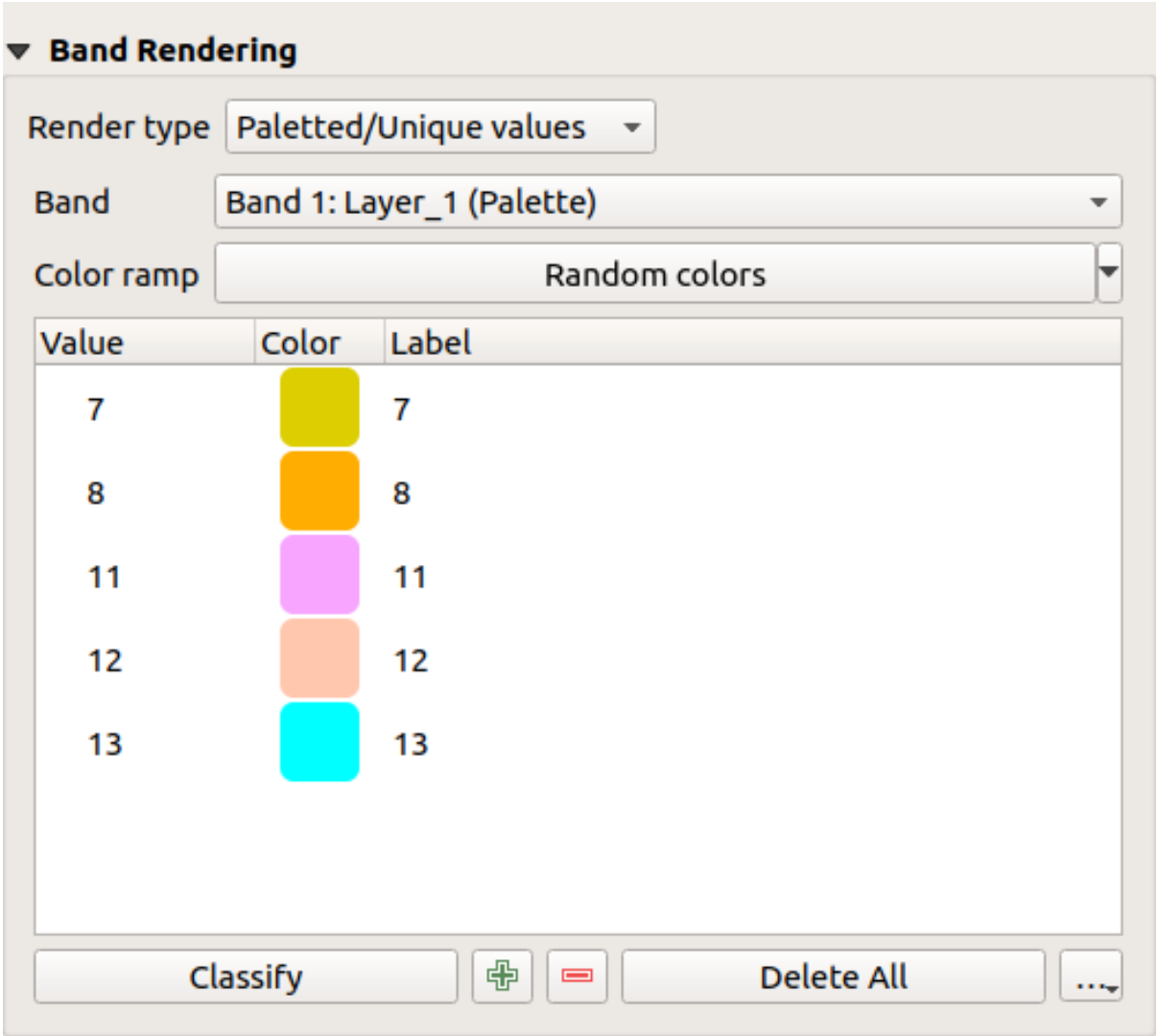


図 17.3: ラスタシンボロジ - カテゴリ値パレットレンダリング

カラーマップの下の右にある ... （詳細オプション）ボタンをクリックして開くと表示されるプルダウンメニューには、カラーマップの読み込み（**ファイルからカラーマップを読み込む**... と書き出し（**ファイルへのカラーマップのエクスポート**... ）、クラスの読み込み（**レイヤからクラスを読み込む**）があります。

単バンドグレー

このレンダラーは、ただ一つのバンドを **グラデーション** :「黒から白」または「白から黒」を使用してレンダリングします。色付けする値の範囲（**最小値（*Min*）** と **最大値（*Max*）** ）は、**最小/最大値設定** で変更できます。

コントラスト は次の値をとります：「強調しない」、「最小最大範囲に引き伸ばす」、「最小最大範囲に引き伸ばしカット」、「最小値から最大値までの範囲以外は無視」

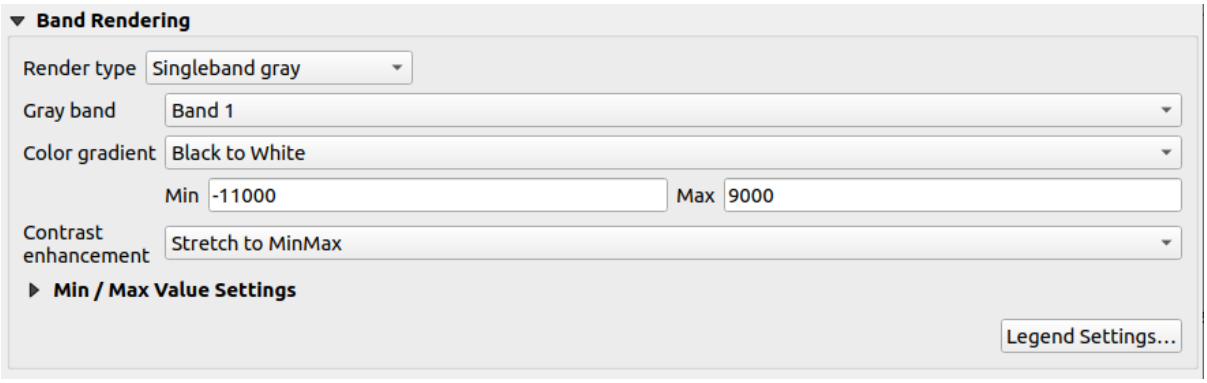


図 17.4: ラスタシンボロジ - 単バンドグレーレンダリング

ピクセルには選択されたカラーグラデーションに基づいた色が割り当てられ、レイヤの凡例（**レイヤ** パネルと印刷レイアウトの **凡例アイテム** ）は連続カラーランプを使用して表示されます。凡例の設定を調整したい場合には、 **凡例の設定**... ボタンを押してください。詳細は **ラスタの凡例のカスタマイズ** を参照してください。

単バンド疑似カラー

これは連続的なパレットを持つ単バンドのファイルに対するレンダラーオプションです。マルチバンドラスタのバンドの一つに対するカラーマップを作成することもできます。

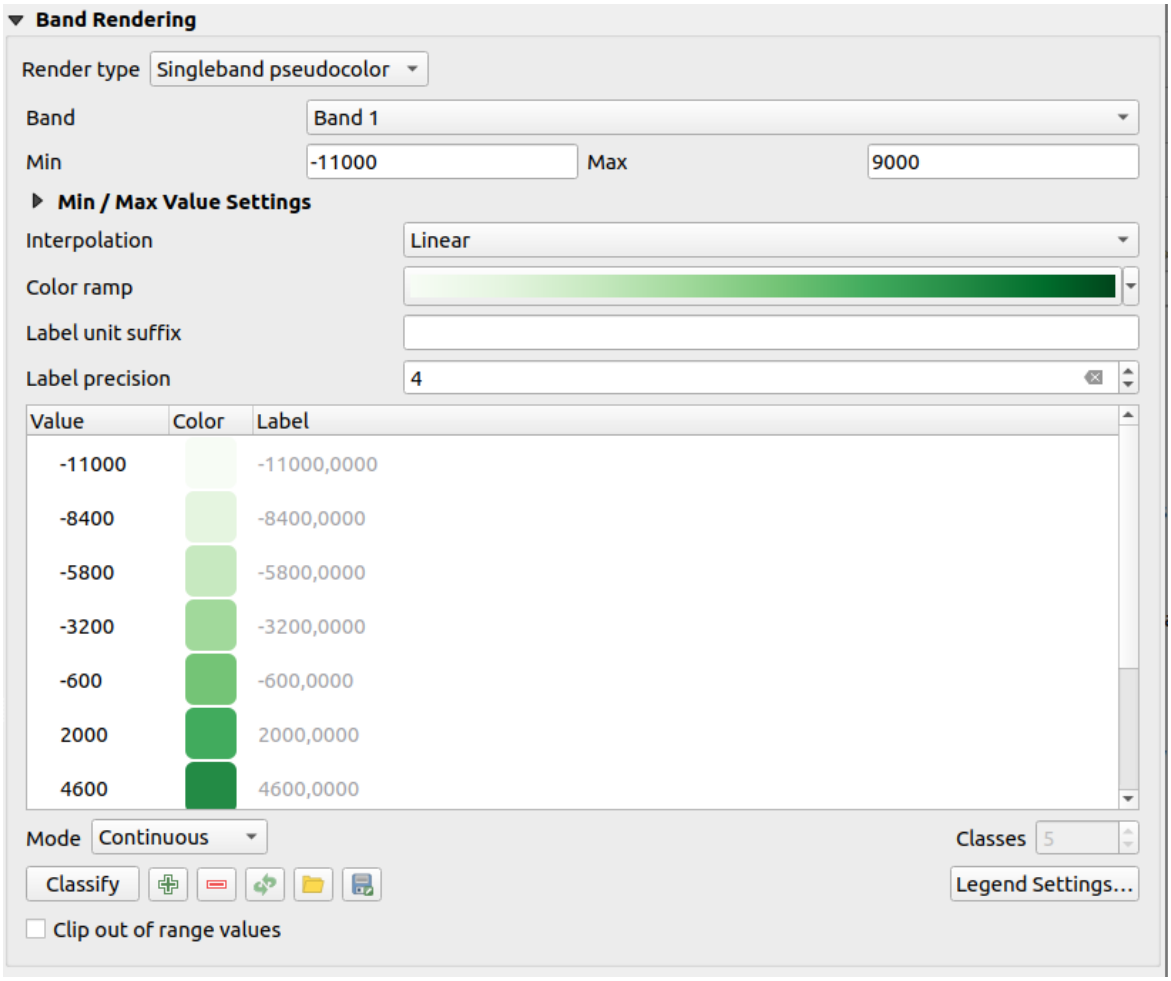


図 17.5: ラスタシンボロジ - 単バンド疑似カラーレンダリング

レイヤの **バンド** と **値の範囲** を使用して、 値の内挿方法とクラス内のピクセルの表現色の割り当てができます。詳細は **カラーランプシェーダの分類** を参照してください。

ピクセルには選択されたカラーランプに基づいた色が割り当てられ、レイヤの凡例（ **レイヤ** パネルと印刷レイアウトの **凡例アイテム** ）は連続カラーランプを使用して表示されます。凡例の設定を調整したり、代わりに個別クラス（と色）を使用した凡例を使いたい場合には、 **凡例の設定**... ボタンを押してください。詳細は **ラスタの凡例のカスタマイズ** を参照してください。

陰影図（hillshade）

ラスタレイヤのバンドを陰影でレンダリングします。

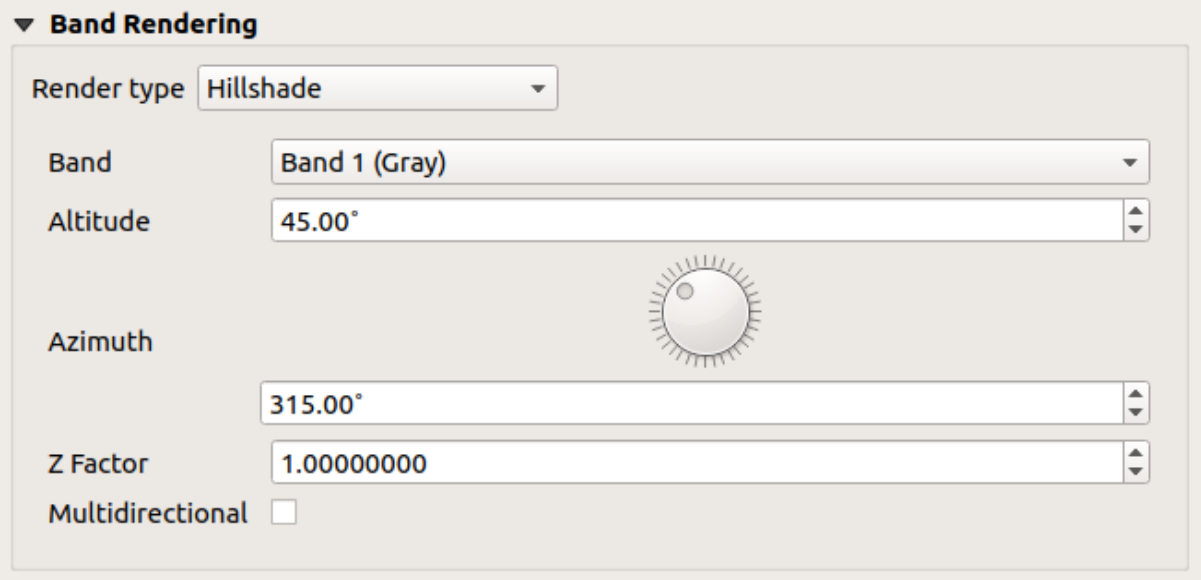


図 17.6: ラスタシンボロジ - 陰影図レンダリング

オプション:

- **バンド** ：使用するラスタのバンド
- **高度** ：光源の仰角（デフォルトは 45 ° ）
- **方位角（*Azimuth*）** ：光源の方位角（デフォルトは 315 ° ）
- *Z* **係数** ：ラスタバンドの値のスケーリング係数（デフォルトは 1 ）
- **多方向シェーディング** ：多方向のシェーディングを利用するかどうかの指定（デフォルトは `off` ）

等高線（Contours）

このレンダラーは、ソースラスタバンドからオンザフライで計算された等高線を描画します。

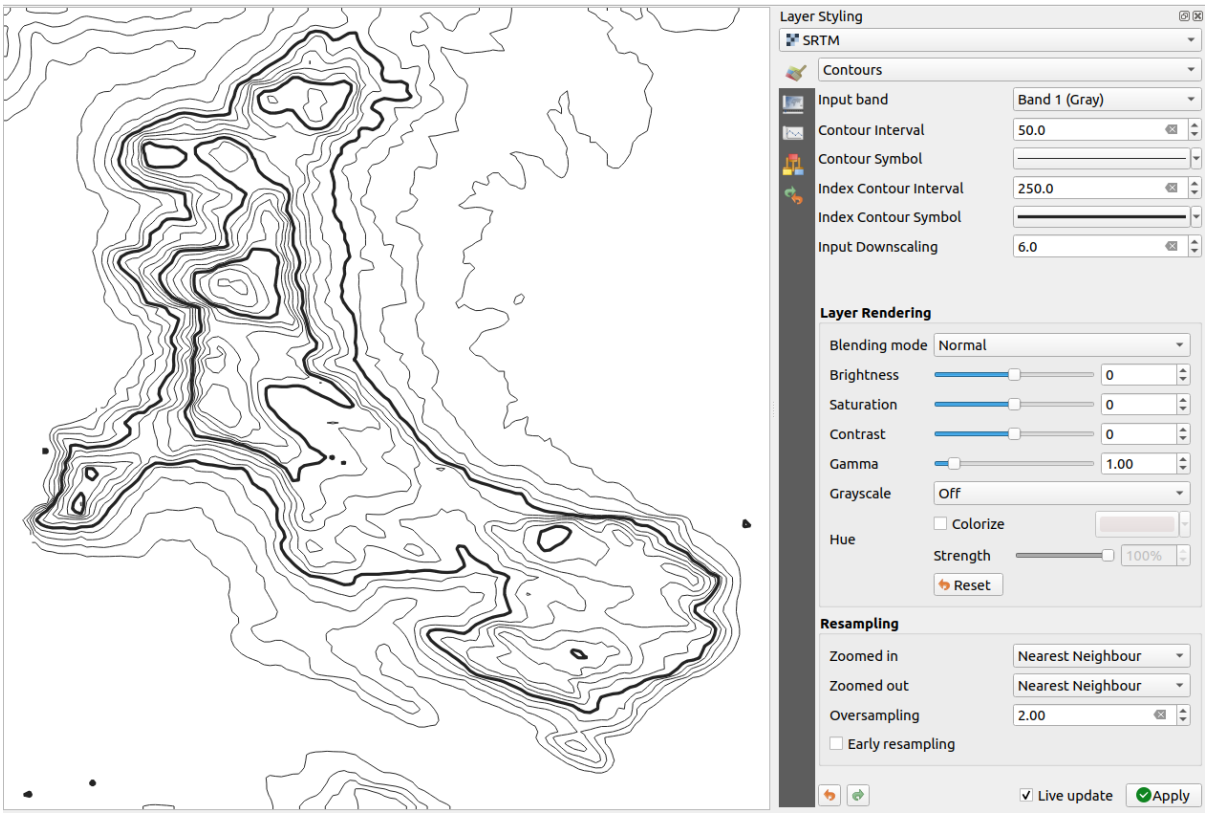


図 17.7: ラスタシンボロジ - 等高線レンダリング

オプション:

- **入力バンド** ：使用するラスタバンド
- **等高線の間隔** ：連続する 2 つの等高線の間隔
- **等高線シンボル** ：一般の等高線に適用する **シンボル**
- **基準等高線の間隔** ：連続する 2 つの **基準等高線** の間隔。基準等高線は識別しやすいように目立たせて表示され、他の等高線よりも太い線で描かれ、線に沿って値のラベルが付いていることが一般的です。
- **基準等高線シンボル** ：基準等高線に適用するシンボル
- **入力値の乗数** ：レンダラーがデータプロバイダへのリクエストをどれだけスケールダウンするかを指定します（デフォルトは `4.0` ）。

  例えば、出力ラスタブロックと同じサイズで入力ラスタブロックの等高線を生成した場合、細かすぎる線が生成されるでしょう。この詳細度合いは、ソースラスタの解像度を下げるようにする「入力値の乗数」によって減らすことができます。1000x500 のラスタブロックでダウンスケール 10 の場合、レンダラーはプロバイダに 100x50 のラスタを要求します。ダウンスケールを大きくすると、等高線がよりシンプルになります（ただし、詳細が失われます）。

最小 / 最大値設定

デフォルトでは、QGIS はラスタのバンドの **最小** 値と **最大** 値を報告します。ごく少数の非常に小さい値や非常に大きい値があると、ラスタのレンダリングに悪影響を及ぼすことがあります。 **最小/最大値設定** フレームはレンダリングの制御に役立ちます。

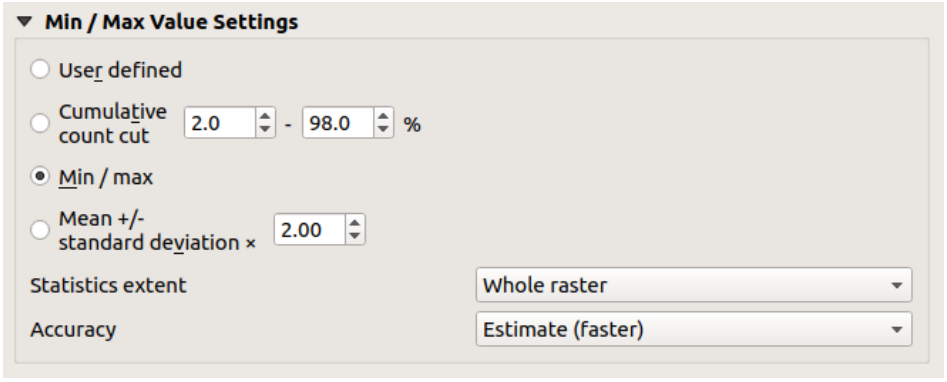


図 17.8: ラスタシンボロジ - 最小 / 最大値設定

利用可能なオプションは次のとおりです。

- **ユーザー定義** ：バンドのデフォルトの **最小** 値や **最大** 値を上書きできます。
- **累積範囲** ：外れ値を除外します。標準の値の範囲は 2% から 98% の間ですが、手動で調整することもできます。
- **最小 / 最大** ：イメージバンドの値の範囲全体を使用します。
- **平均** +/- **標準偏差** x ：標準偏差の範囲内、もしくは標準偏差の倍数の範囲内の値のみを考慮したカラーテーブルを作成します。これは、ラスタレイヤ内に異常に大きな値を持ったセルが 1 つ 2 つあり、これがラスタのレンダリングに悪影響を与える場合において便利です。

バンドの最小値・最大値の計算は以下に基づいて行われます：

- **集計する範囲** ：これは **ラスタ全体** 、 **現在のキャンバス** または **更新されたキャンバス** の値をとります。 **更新されたキャンバス** は、レンダリングに使用する最小値/最大値がキャンバス範囲に応じて変わる（動的引き伸ばし）ことを意味します。
- **精度** ： **推定値（高速）** または **実際の値（低速）** のどちらかを指定します。

注釈: 一部の設定では、実際の最小値と最大値をウィジェット内に表示するためにレイヤプロパティダイアログの **適用** ボタンを押す必要があります。

カラーランプシェーダの分類

これは、スカラデータセット（ラスタやメッシュのコンター）をその値に基づいて分類し表現するために利用されます。 **カラーランプ** とクラス数が与えられると、このクラス数に対する中間的なカラーマップエントリが生成されます。値の範囲から補間された値と分類モードに応じた値で各色がマッピングされます。スカラデータセットの要素には、そのクラスに基づいて色が割り当てられます。

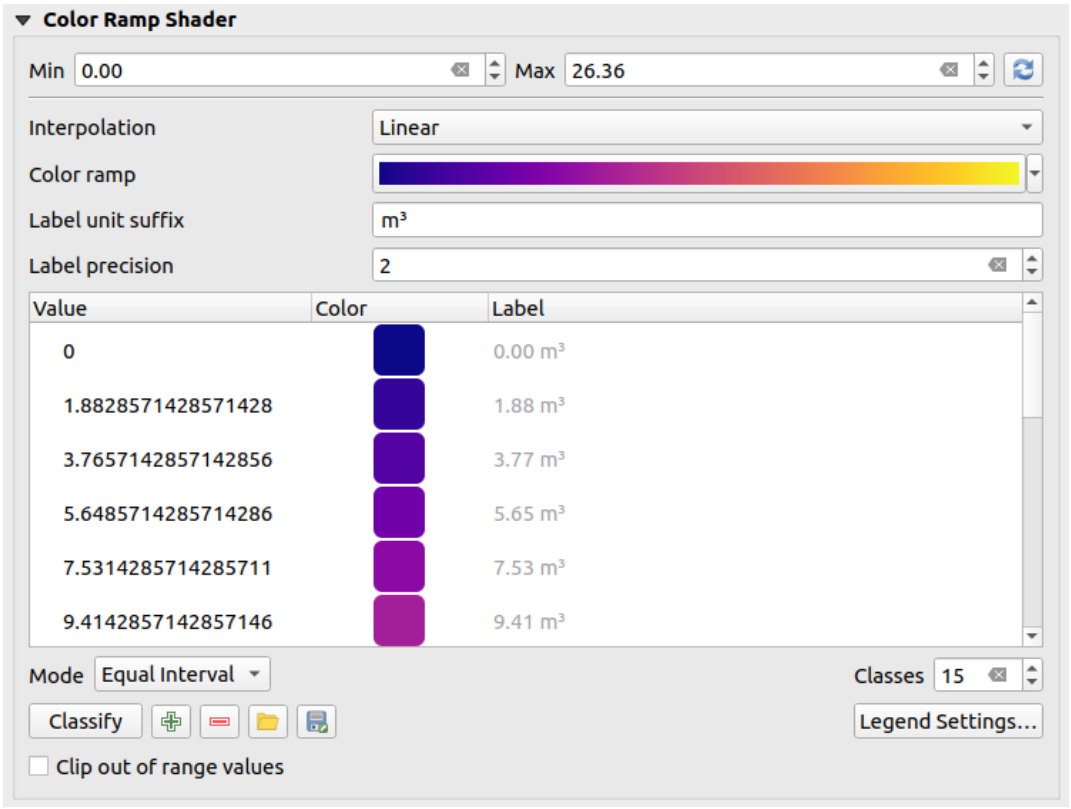


図 17.9: カラーランプシェーダを使用したデータセットのクラス分類

1. **最小値（*Min*）** と **最大値（*Max*）** が定義されている必要があり、これがクラス境界の内挿に使用されます。QGIS はデフォルトでデータセットから最小値・最大値を検出しますが、修正することも可能です。

2. **内挿** エントリは、スカラー要素にどのように色に割り当てるかを定義します：

   - **離散的** （<= シンボルが **値** カラムのヘッダに表示されます）：ラスタの色には、その値以上で最も近いカラーマップのエントリを使用します。
   - **線形** ：ラスタの色は、ピクセル値の上下のカラーマップエントリから線形に補間されます。これにより、データセットの各値には固有の色が対応します。
   - **正確な** （= シンボルが **値** カラムのヘッダに表示されます）：カラーマップエントリの値に等しい値を持つピクセルだけに色が適用されます。その他の値のピクセルはレンダリングされません。

3. **カラーランプ** ウィジェットによって、データセットに割り当てるカラーランプを選択できます。 **このウィジェット** と同様に、新しいカラーランプを作成したり、現在選択されているカラーランプを編集したり保存したりすることができます。カラーランプの名前は設定に保存されます。

4. **ラベルの単位の接尾辞** は、凡例内で値の後ろにラベルを追加します。また、 **ラベルの精度** は表示す

る小数点以下の桁数を制御します。

5. 分類の **モード** は、クラス間でどのように値が分布するかを定義できます：

   - **等間隔分類** ： **クラス** の数を指定すると、全てのクラスの間隔が同じ大きさとなるように値が定義されます。
   - **連続的** ：クラス数と色はカラーランプのグラデーションストップから取得します。値はカラーランプのグラデーションストップの分布に応じて設定されます。
   - **等量分類** ： **クラス** の数を指定すると、クラスが同じ要素数となるように値が定義されます。これは **メッシュレイヤ** では利用できません。

6. 次に、 **分類** ボタンを押すか、以下の方法でクラスを微調整します：

   -  手動で値を追加 ボタンはテーブルに値を追加します。
   -  選択した行を削除 ボタンはテーブルから選択した値を削除します。
   - **値（*Value*）** カラムをダブルクリックするとクラス値を編集できます。
   - **色** カラムをダブルクリックすると **色の選択** ダイアログが開き、値に適用する色を選択できます。
   - **ラベル** カラムをダブルクリックするとクラスのラベルを編集できます。ただし、これは地物情報表示ツールを使用した場合には表示されません。
   - 色テーブルの選択した行で右クリックすると、選択行に対する **色を変更**... と **不透明度を変更**... のコンテキストメニューが表示されます。

    ファイルからカラーマップをロード ボタンや  カラーマップをファイルにエクスポート ボタンを使用して、既存のカラーテーブルを読み込んだり、後で使用するためにカラーテーブルを保存したりできます。

7. **内挿** で「線形」オプションを選択した場合は、以下の設定もできます：

   -  **範囲外の値を無視** ：デフォルトでは、線形内挿は **最小値** よりも小さいデータセットの値を最初のクラスの色（同様に **最大値** よりも大きな値を最後のクラスの色）に割り当てます。最小値・最大値より外の値をレンダリングしたくない場合には、この設定にチェックを入れてください。
   - **凡例の設定**... ： **レイヤ** パネルや印刷レイアウトの **凡例アイテム** の表示のためのオプションです。詳細は **ラスタの凡例のカスタマイズ** を参照してください。

ラスタの凡例のカスタマイズ

ラスタレイヤやメッシュレイヤにカラーランプを適用する場合、凡例をクラス分けして表示したいことがあります。デフォルトでは、QGIS は **レイヤ** パネルや印刷レイアウトの **凡例アイテム** には最大値・最小値付きの連続的なカラーランプを表示します。この凡例は、クラス分類ウィジェットの **凡例の設定**... ボタンを使用してカスタマイズすることができます。

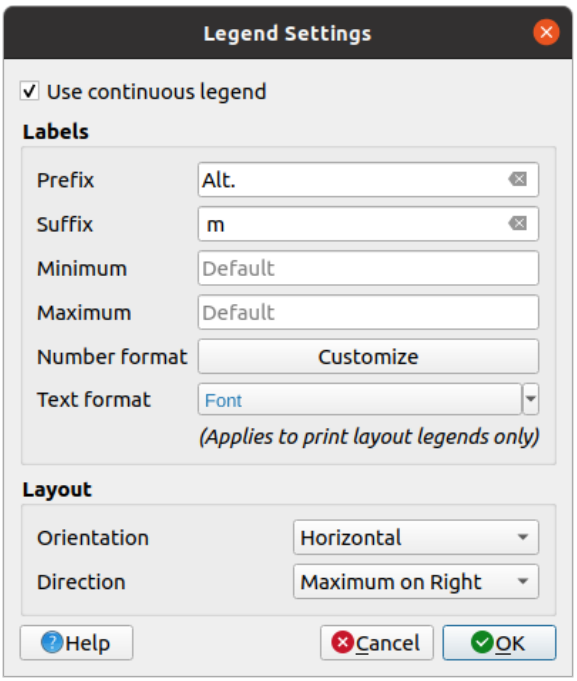


図 17.10: ラスタの凡例の修正

このダイアログでは、**表示に連動する凡例を使用する** かどうかを設定できます。チェックを外すと、凡例はさまざまな階級に対応して適用されるバラバラの色で表示されます。このオプションは、**単バンドグレー** シンボロジのラスタでは利用できません。

表示に連動する凡例を使用する にチェックを入れると、凡例のラベル、レイアウトのプロパティの両方を設定できます。

ラベル

- ラベルへの **接頭辞** や **単位文字** の追加
- 凡例に表示する **最小値** や **最大** 値の修正
- **数値フォーマット** の **カスタマイズ**
- 印刷レイアウトの凡例に使用する **テキストフォーマット** の **カスタマイズ**

レイアウト

- 凡例カラーランプの **方向** の設定： **垂直** または **水平**
- 上記の方向に応じた、値の **方向** の設定
  - 方向が垂直の場合には、 **上が最大** または **上が最小**
  - 方向が水平の場合には、 **右が最大** または **右が最小**

レイヤレンダリング

レイヤのバンドに適用するシンボロジタイプを設定できたら、ラスタファイル全体に適用される特別なレンダリング効果を与えましょう。

- 混合モードのいずれかを使用する（ **混合モード** 参照）
- 色の **輝度** 、 **彩度** 、 **ガンマ** 、 **コントラスト** をカスタマイズする
- **カラーを反転** を使用して、レイヤを反対色でレンダリングする。例えば、OpenStreetMap のタイルをダークモードにさっと切り替えるのに便利です。
- レイヤを **グレースケール** に変換する。オプションには「明度を使用」、「彩度を使用」、「平均を使用」があります。
- **着色** を使用して、カラーテーブルの **色相** の **強調** を調整する

リセット ボタンを押すと、レイヤレンダリングのカスタム変更をすべて削除します。

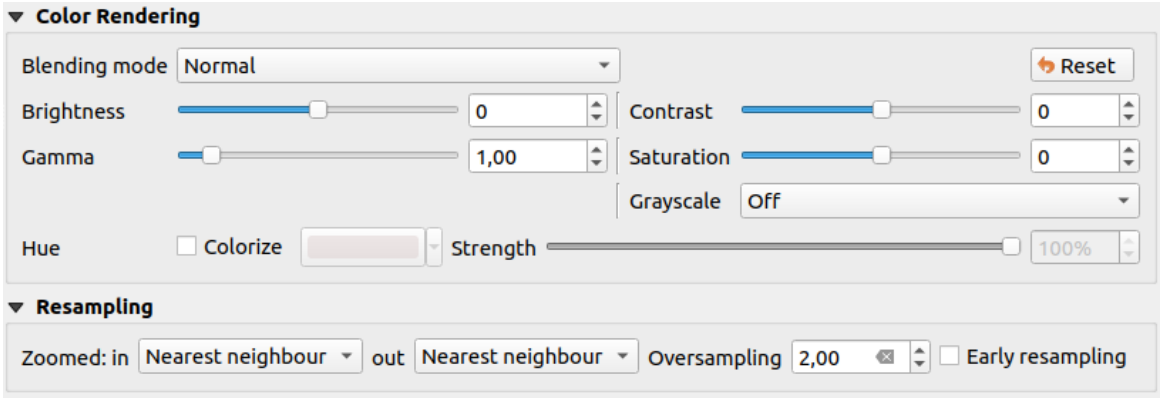


図 17.11: ラスタシンボロジ - レイヤレンダリング設定およびリサンプリング設定

リサンプリング

リサンプリング オプションは、画像の拡大・縮小時に影響を与えます。リサンプリングモードは地図の見た目を最適化することができます。これは、幾何学的変換によって新しいグレー値行列を計算します。

「最近傍（Nearest neighbour）」法を適用する場合、地図を拡大していくとピクセル化された構造となることがあります。このような見た目は、「バイリニア（Bilinear）」法もしくは「キュービック（Cubic）」法を使用すると改善することができます。これらの手法はシャープな縁をぼかして、結果として滑らかな画像が得られます。この手法は、例えばデジタル地形図のラスタ地図などに適用すると良いでしょう。

Early resampling: ソースの解像度がわかっているプロバイダレベルでラスターレンダリングを計算することができ、QGIS カスタムスタイルでのレンダリングでより良いズームが確保されます。ref:`interpretation method `を使用して読み込まれたタイルラスターにとても便利です。

17.1.4 透過性プロパティ

QGIS には、ラスタレイヤの透過性レベルの設定機能があります。

グローバルな不透明度 スライダーを使用すると、（もしあれば）現在のラスタレイヤの下にあるレイヤがどの程度見えるかを設定できます。これは、ラスタレイヤを重ね合わせる場合（例えば陰影図の上に分類されたラスタ地図を重ねるなど）に非常に便利です。これによって、地図がより立体的に見えるようになります。ラスタの不透明度はデータ定義とすることもでき、例えば他のレイヤの可視性や時間変数、地図帳のページの違いに応じて変化させることができます。

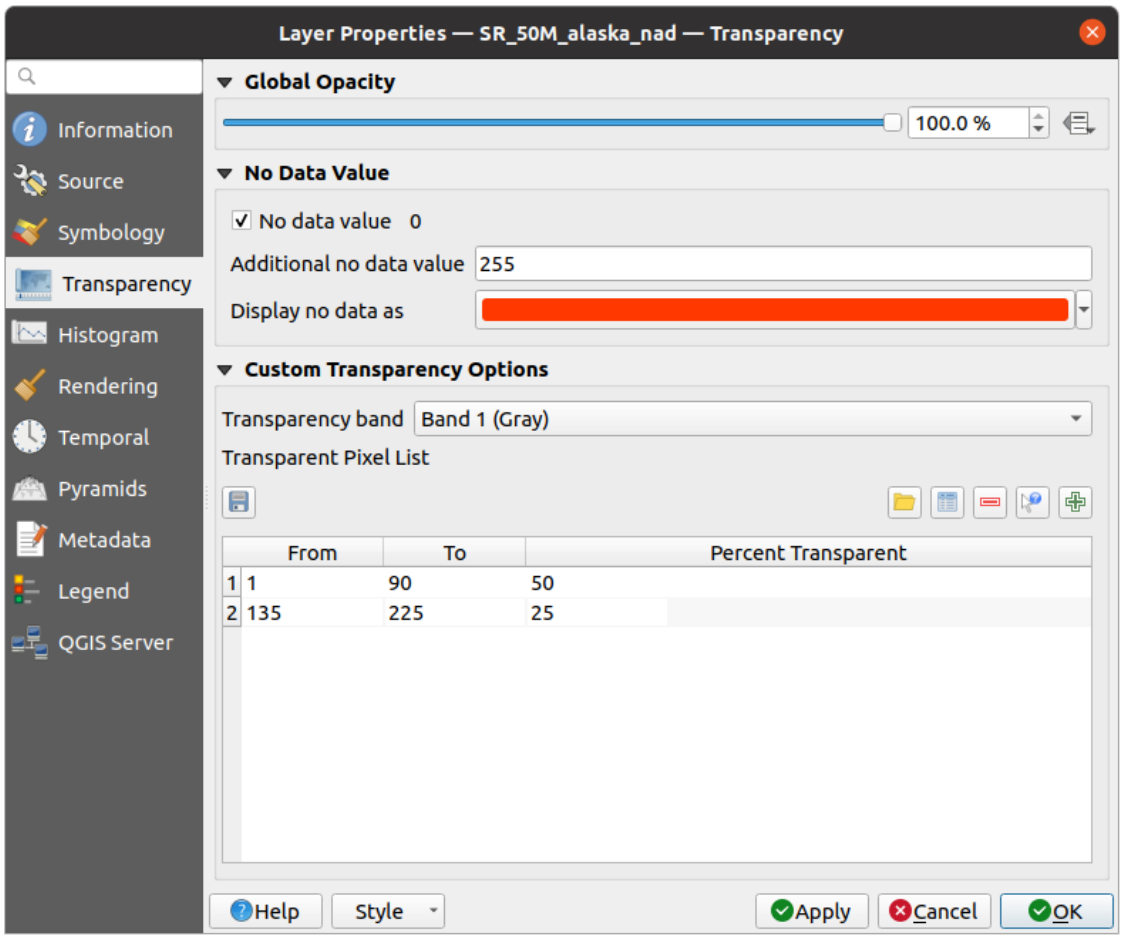


図 17.12: ラスタの「透過性」タブ

データなし（*nodata*）とする値 には、QGIS が取得した元のソースの nodata 値が（もし定義されていれば）表示されます。この値は、レンダリング時にデータなしと見なされます。さらに、**追加の *nodata* 値** として扱われるラスタ値を入力することもできます。 ***no data* の代替表示** のカラーセレクタを使用すると、デフォルトの透明なレンダリングの代わりに、自分で設定した色を nodata 値のピクセルに適用できます。

さらに自由度の高い透過性のカスタマイズが **カスタム透過オプション** のセクションで利用可能です：

- **透過設定するバンド** を使用して、バンド全体に不透明度を適用します。
- 透過させるピクセルのリストを対応する透過率とともに指定します。
  1. 手動で値を追加 ボタンをクリックします。新しい行がピクセルリストに表示されます。

2. **赤**、**緑**、**青** のピクセル値を入力し、適用する **透過率** を調整します。
3. この代わりに、  画面から値を追加 ボタンを使用して、ラスタから直接ピクセル値を取得することもできます。透過率の値は入力してください。
4. この手順を繰り返し、さらに多くの値のカスタム透過率を調整します。
5. **適用** ボタンを押して、地図を確認してみましょう。

このように、カスタム透明度を設定するのはとても簡単ですが、かなりの手間がかかります。そのため、  ファイルにエクスポート ボタンを使用して、透明度リストをファイルに保存できます。  ファイルからインポート ボタンは、透明度の設定をロードして現在のラスタレイヤに適用します。

17.1.5 ヒストグラムプロパティ

 **ヒストグラム** タブでは、ラスタ内の値の分布を確認できます。ヒストグラムは、**ヒストグラムの計算** ボタンを押すと生成されます。既存のバンドは全て一緒に表示されます。ヒストグラムを画像として保存するには、  ボタンを使用します。

ヒストグラムの下部では、ドロップダウンメニューのラスタバンドを選択して、**最小/最大のスタイルを設定** することができます。  **設定/アクション** ドロップダウンメニューには、ヒストグラムをカスタマイズするための高度なオプションがあります。

- **可視性** オプションを使用すると、個別バンドごとにヒストグラムを表示できます。これには  **選択したバンドを表示する** オプションを選択してください。
- **最小/最大オプション** では、「最小/最大マーカーを常時表示」、「最小/最大にズーム」、「最小/最大にスタイルを更新」の設定ができます。
- **アクション** オプションでは、バンドの最大・最小値を変更した後で「リセット」や「ヒストグラムを再計算」することができます。

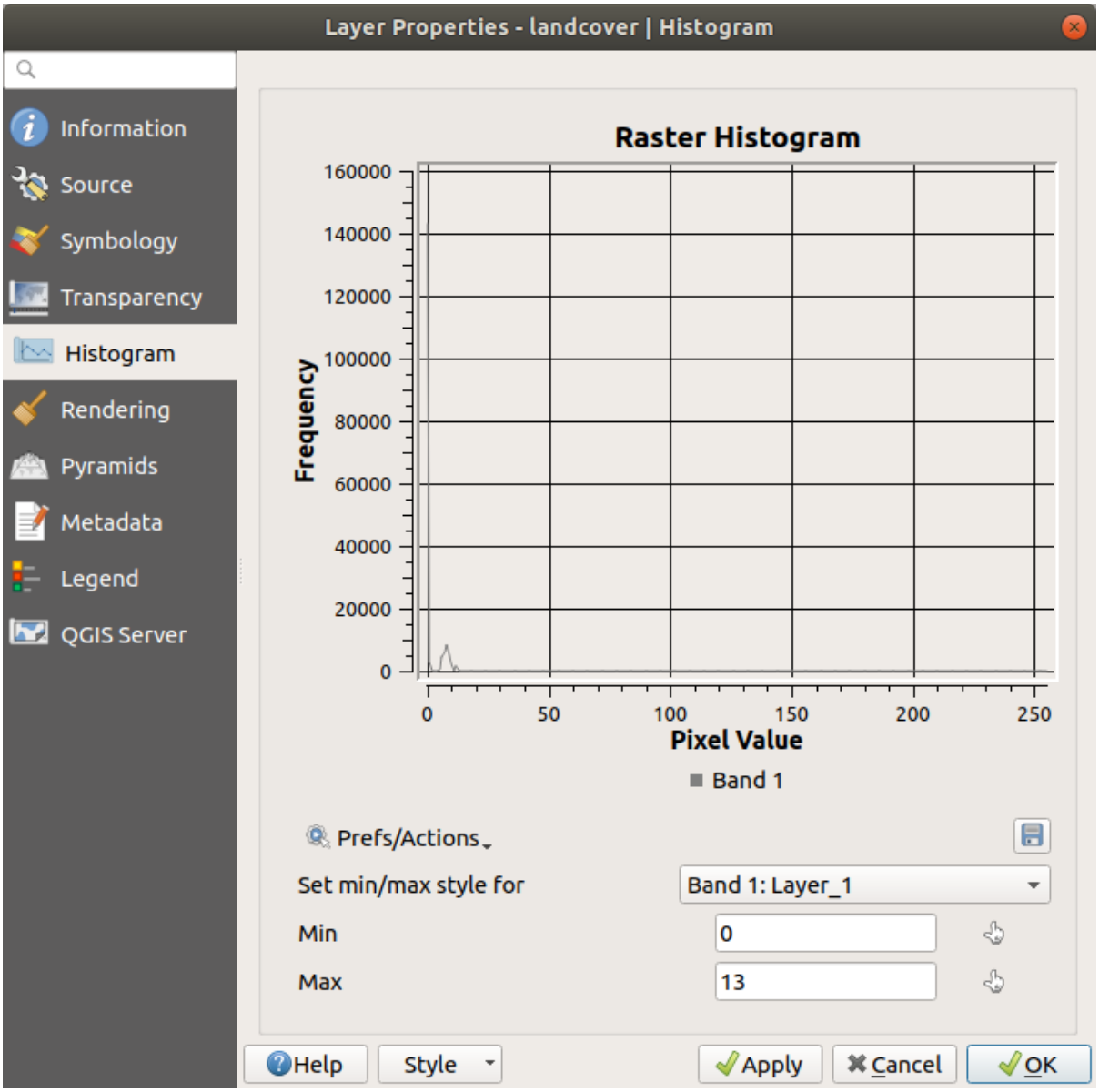


図 17.13: ラスタの「ヒス　　トグラム」タブ

17.1.6 レンダリングプロパティ

レンダリング タブでは、以下の設定が可能です：

- レイヤの **縮尺に応じた表示設定** ： **最大縮尺（この値を含む）** と **最小縮尺（この値を含まない）** を設定して、地物が表示される縮尺の範囲を定義できます。この範囲の外では、地物は非表示になります。 現在のキャンバスの縮尺に設定 ボタンを使用すると、現在のマップキャンバスの縮尺を境界値として使用できます。詳細については、 **表示縮尺セレクタ** を参照してください。

注釈: レイヤの縮尺に応じた表示は、**レイヤ** パネルからも有効にすることもできます: レイヤを右

クリックし、コンテキストメニューから **レイヤの縮尺表示を設定** を選択します。

- **再描画間隔（秒）** ：タイマーを設定して、個々のレイヤを自動的に更新します。複数のレイヤに自動更新間隔が設定されている場合、複数回の更新を避けるためにキャンバスの更新は延期されます。

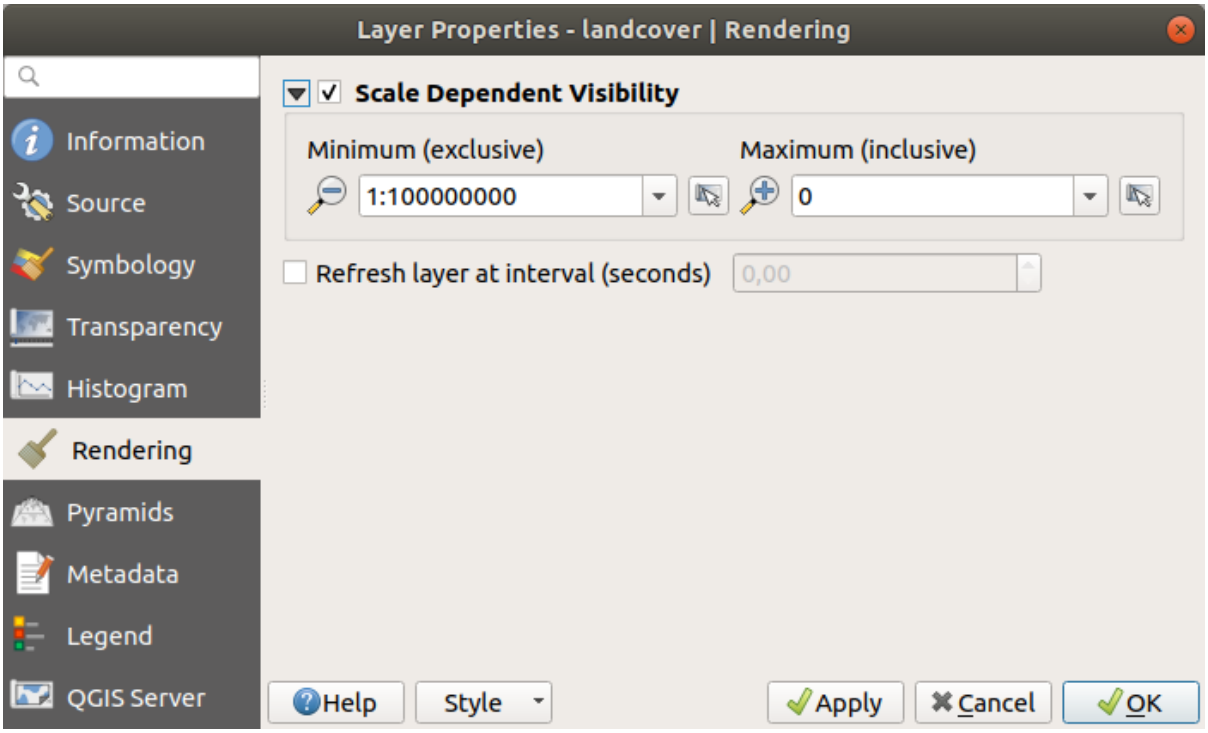


図 17.14: ラスタのレンダリングプロパティ

17.1.7 時系列プロパティ

時系列 タブは、時間経過に伴うレイヤのレンダリングを制御するオプションを提供します。このような動的なレンダリングには、マップキャンバスで **時系列ナビ** を有効にする必要があります。

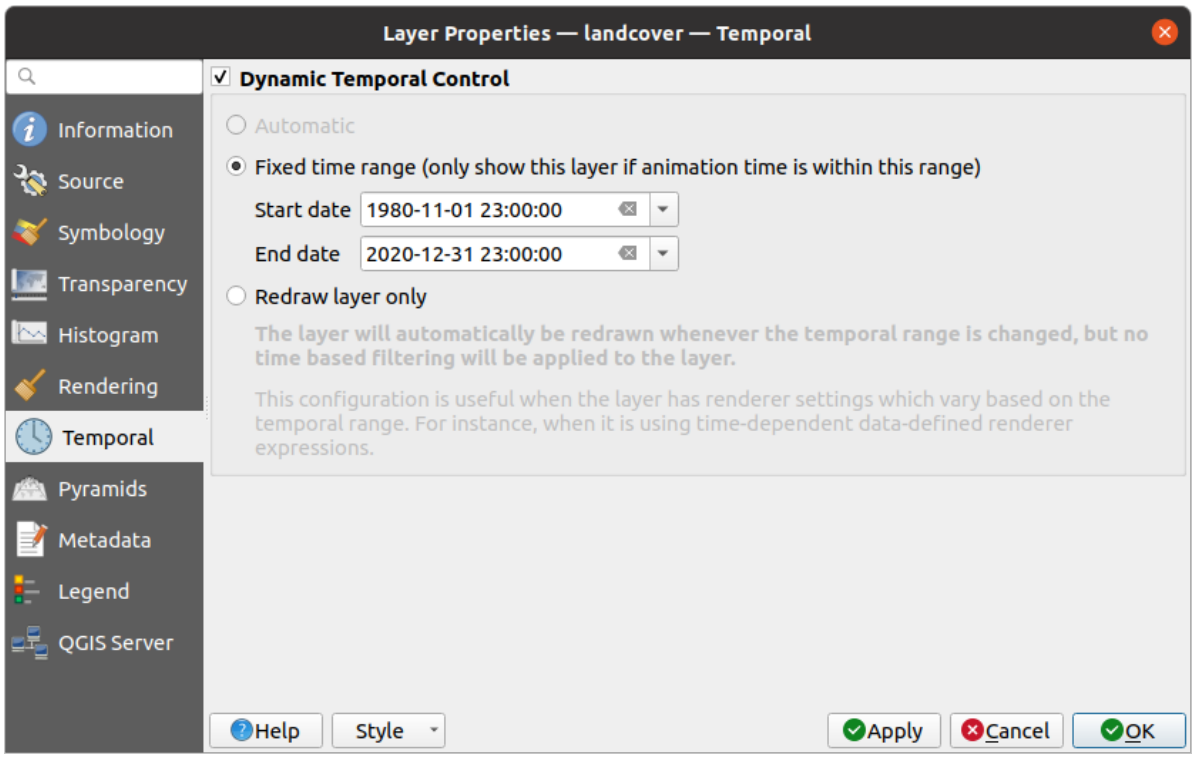


図 17.15: ラスタの時系列プロパティ

動的時系列コントロール オプションにチェックを入れ、レイヤの再描画をどのようにするか設定します：

- **自動** ：元のデータプロバイダが時系列データの処理をサポートしている場合、レンダリングは元のデータプロバイダによって制御されます。この設定は例えば、WMS-T レイヤや、PostGIS ラスタレイヤ等で利用できます。
- **固定時間範囲** ：アニメーション時刻が **開始時刻** と **終了時刻** の範囲内にある場合にのみ、このラスタレイヤを表示します。
- **レイヤのみ再描画** ：レイヤは新しいアニメーションフレーム毎に再描画されます。これは、レンダラーの設定に時間ベースの式の値を使用しているレイヤ（例：レンダラーの透明度をデータによって定義することで、ラスタレイヤをフェードイン/アウトする）に便利です。

17.1.8 標高プロパティ

標高 タブには、 *3D* **マップビュー** 内のレイヤの標高プロパティと、 *profile tool charts* 内のレイヤの標高をコントロールするオプションがあります。具体的には次が設定できます:

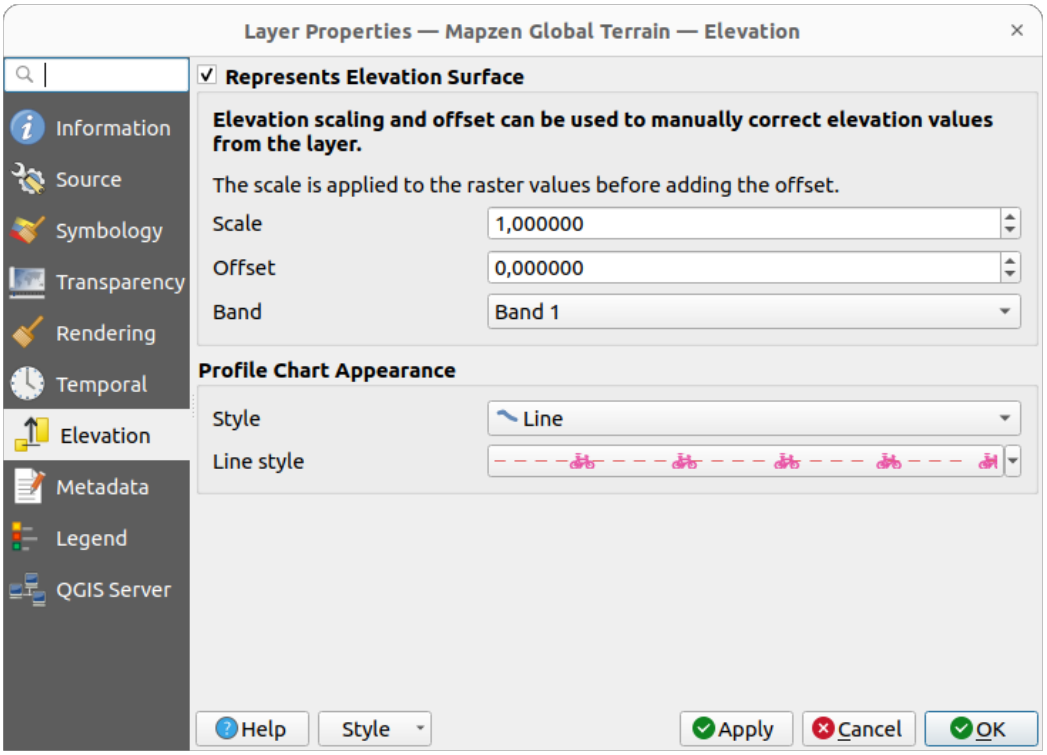


図 17.16: ラスタ標高プロパティ

- **標高サーフェスを表示**: ラスタレイヤが標高面（DEM など）を表し、ピクセル値が標高として解釈されるかどうかを指定します。ラスタを **標高断面図** で表示したい場合は、このオプションをチェックします。また、値を拾う **バンド** に入力する必要があり、 **スケール** 係数と **オフセット** を適用することができます。
- **断面図の外観**: 断面図を描く際に、ラスター標高が使用するレンダリングの **スタイル** を制御します。次のように設定できます：
  - a profile *Line* with a *line style* applied
  - **下を塗りつぶす** と対応する **塗りつぶしスタイル** を持つ表面

17.1.9 ピラミッドプロパティ

QGIS で高解像度のラスタレイヤを使用すると、操作が遅くなることがあります。データの低解像度のコピー（ピラミッド）を作成することで、QGIS はズームレベルに応じて使用する最適な解像度を選択するため、パフォーマンスが大幅に向上します。

ピラミッドを作成するには、オリジナル画像があるディレクトリへの書き込み権限を持っている必要があります。

解像度 リストからクリックしてピラミッドレベルを作成したい解像度を選択します。

全体図の形式 ドロップダウンメニューから **内部（可能であれば）** を選択すると、QGIS は内部的にピラミッドを構築しようとします。

注釈: ピラミッドの作成は元のデータファイルを変更する可能性があり、一度作成したピラミッドは削除することはできませんのでご注意ください。「ピラミッドのない」ラスタをとっておきたい場合には、ピラミッドを作成する前にバックアップコピーを作成してください。

外部 や **外部（Erdas Imagine）** を選択した場合には、ピラミッドは元のラスタと同じ場所に、同じ名前の `.ovr` 拡張子で作成されます。

ピラミッド作成にはいくつかの **リサンプリング方法** が使用可能です：

- 最近傍
- 平均
- ガウス
- キュービック
- キュービック・スプライン
- ランチョス法
- モード
- なし

最後に **ピラミッドの構築** ボタンを押すと、処理を開始します。

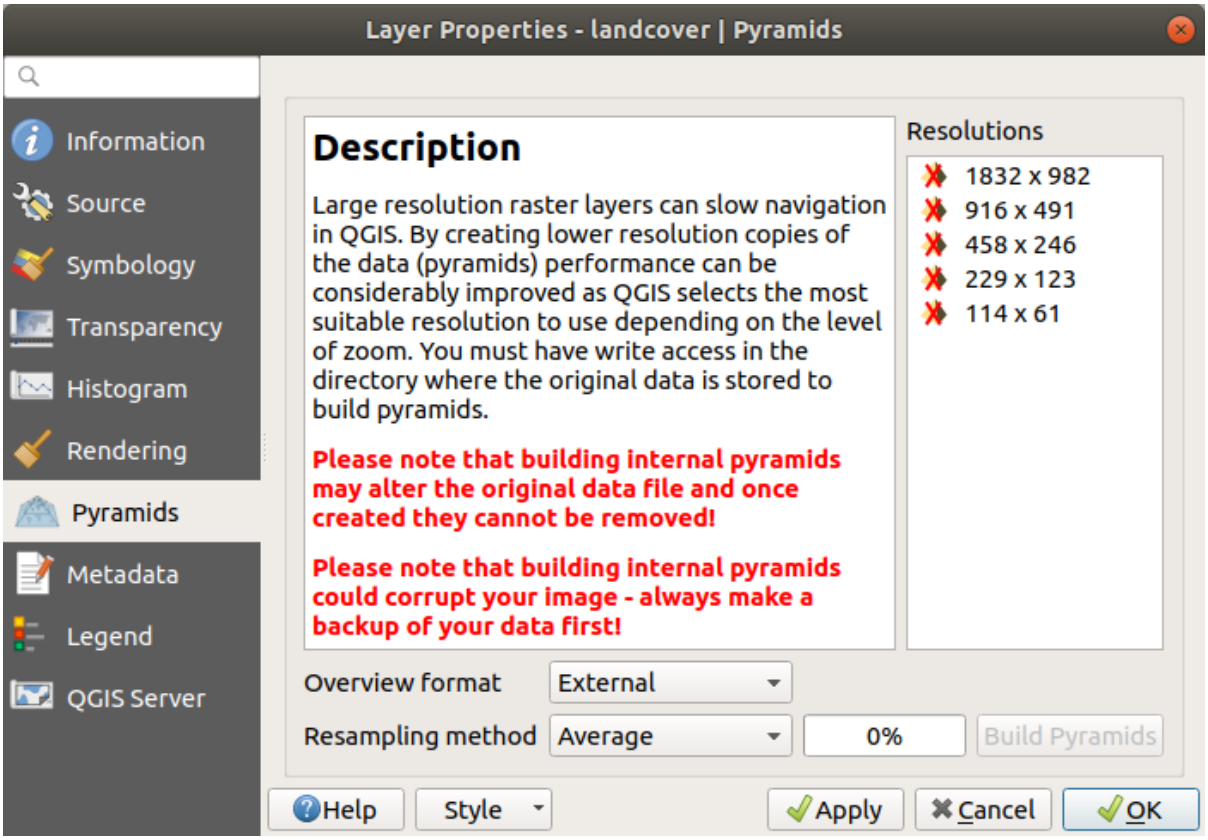


図 17.17: ラスタの「ピラミッド」タブ

17.1.10 メタデータプロパティ

メタデータ タブには、レイヤに関するメタデータレポートの作成・編集オプションがあります。詳細は **メタデータ** を参照してください。

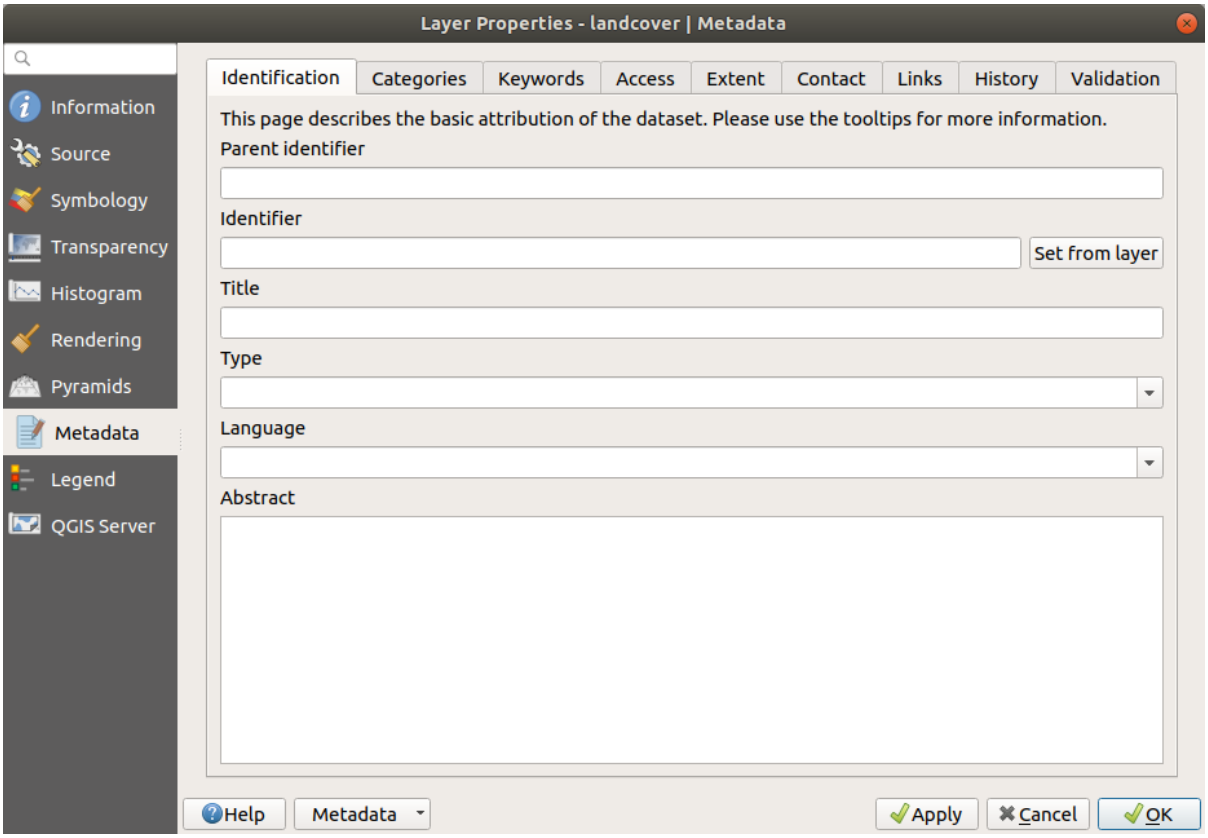


図 17.18: ラスタの「メタデータ」タブ

17.1.11 凡例プロパティ

凡例 タブでは、 **レイヤパネル** や **印刷レイアウトの凡例** に関する高度な設定ができます。これらのオプションには以下のものがあります：

- レイヤに適用しているシンボロジにもよりますが、凡例に複数のエントリが表示されることで、必ずしも読みやすく便利な表示になるとは限りません。 **凡例の枠の画像 `で代わりとなる** *:ref:* **画像の選択** *<embedded_file_selector>* を行うことができ、 **レイヤ** パネルや印刷レイアウトの凡例の両者で表示されます。
- **凡例の埋め込みウィジェット** は、レイヤパネル内のレイヤツリーに埋め込み可能なウィジェットのリストです。このアイディアは、レイヤで頻繁に使用するいくつかのアクション（不透明度の設定、フィルタ、選択、スタイルなど）に簡単にアクセスするための方法を提供することです。

  デフォルトでは QGIS は不透明度ウィジェットを提供していますが、これは独自ウィジェットを登録するプラグインによって拡張することが可能で、管理するレイヤにカスタムアクションを割り当てられます。

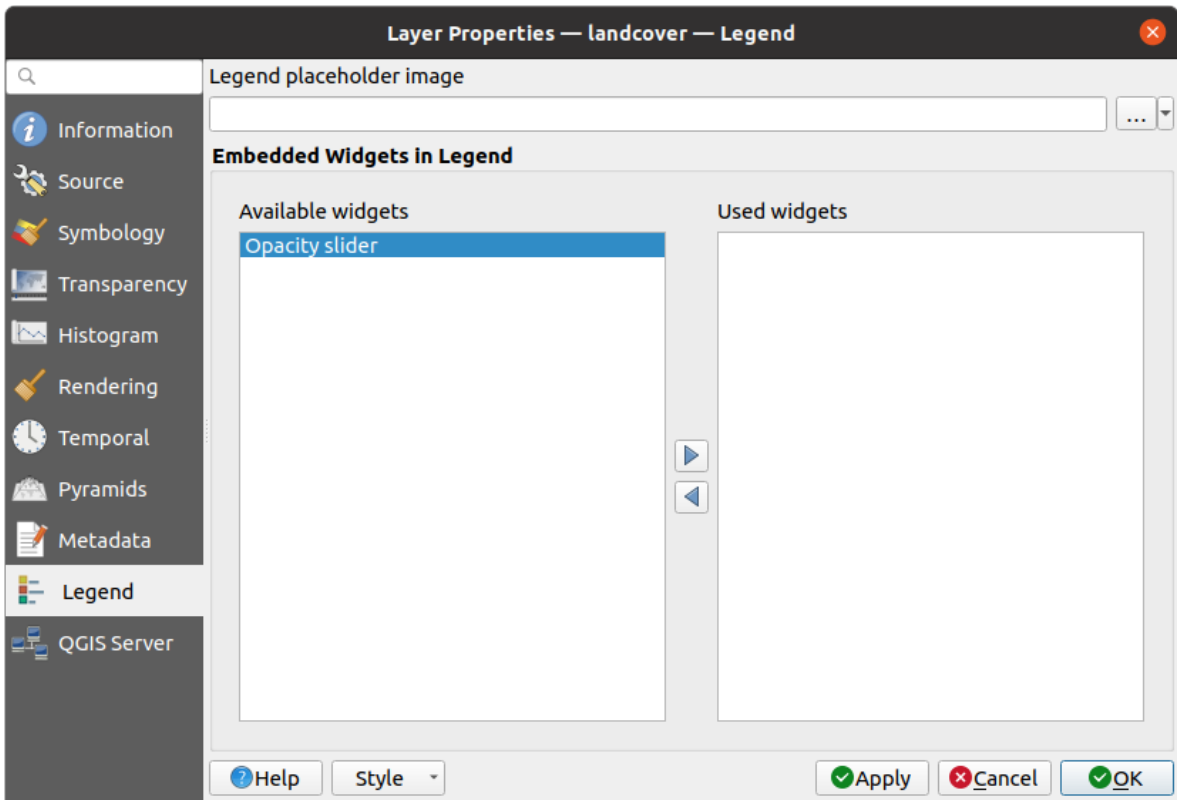


図 17.19: ラスタの「凡例」タブ

17.1.12 QGIS サーバー

QGIS **サーバー** タブでは、 QGIS サーバー で公開されるデータの設定を行うことができます。設定内容は以下の通りです：

- **説明**: **短い名前**、**タイトル**、**要約**, **キーワード**, **形式** が `text/html`、`text/plain` または `application/pdf` である、**データ** *URL* など、データを表す情報を提示します。
- **帰属**: データの提供者を特定するための **タイトル** と *URL*
- **メタデータ** *URL*: `FGDC` または `TC211` **データ型** で、 `text/plain` または `text/xml` **形式** であるメタデータの *URL* 一覧
- **凡例** *URL*: 凡例の *URL* で、`image/png` または `image/jpeg` **形式** のいずれか

注釈: 公開したいラスターレイヤーが既に Web サービスで提供されている場合、さらに *properties* を設定することができます。

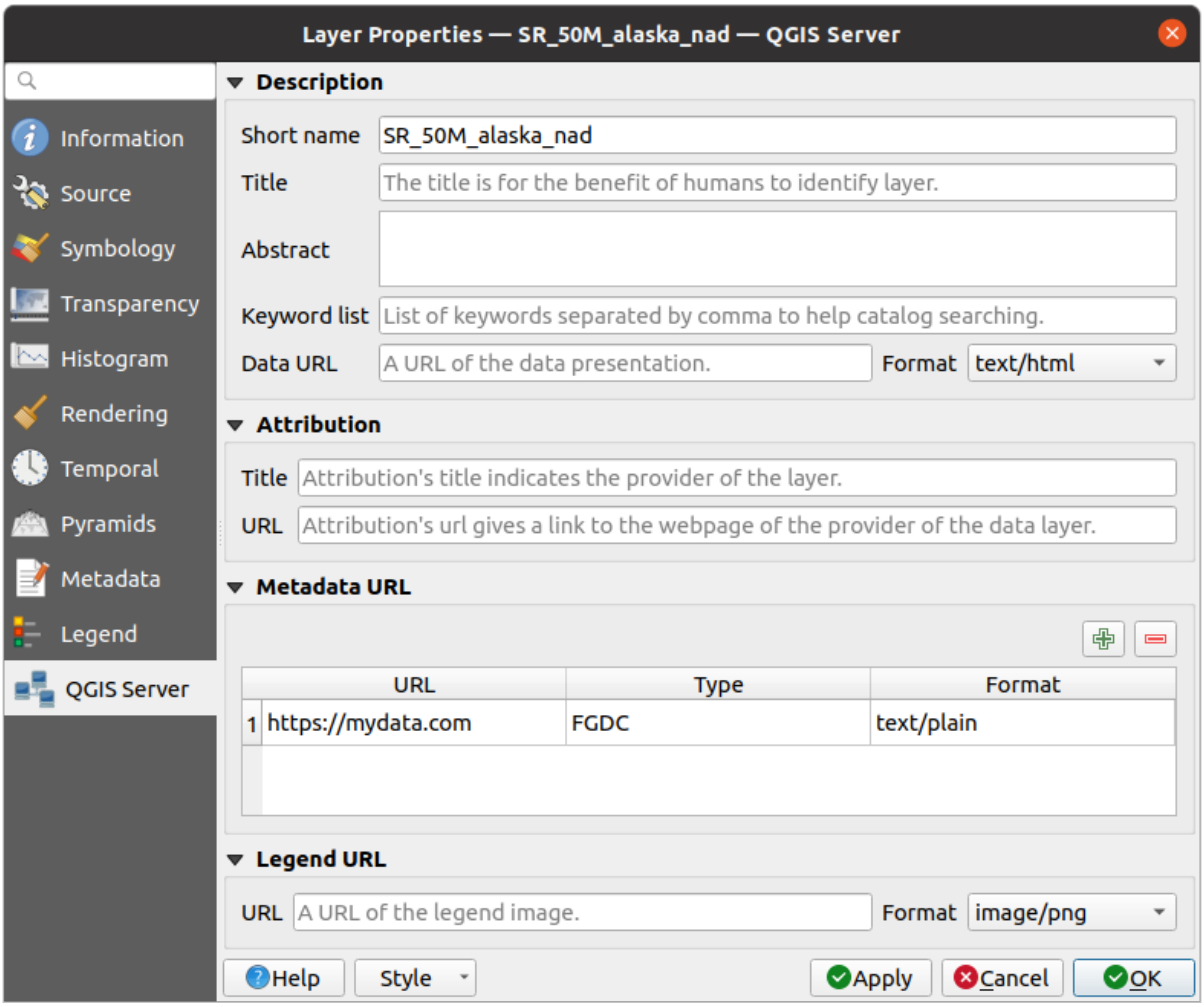


図 17.20: ラスタプロパティの「QGIS サーバー」タブ

17.2 ラスタ解析

17.2.1 ラスタ計算機

ラスタ メニューの **ラスタ計算機** は、既存のラスタのピクセル値に基づいた計算を実行します（ 図 17.21 参照）。結果は GDAL がサポートする形式の新規ラスタレイヤとして書き出されます。

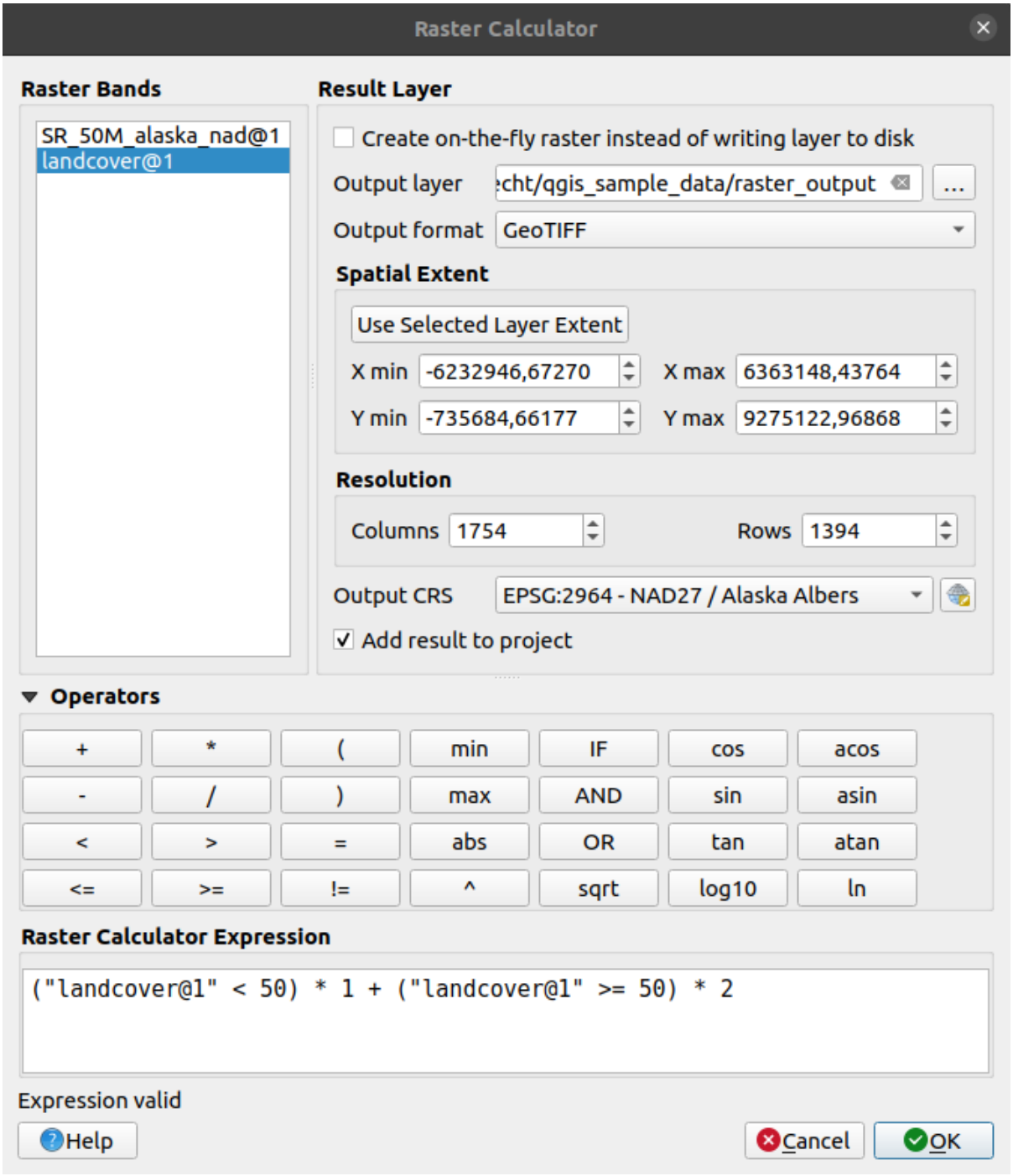


図 17.21: ラスタ計算機

バンド のリストには、使用可能なロードされたラスタレイヤがすべて含まれています。ラスタ計算機の式フィールドにラスタを追加するには、フィールドリスト内で名前をダブルクリックします。そこに演算子ボタンを使用したりボックス内にキーボードで演算子を入力して、計算式を作成します。

ラスタレイヤ セクションでは、出力レイヤを定義する必要があります。以下の設定が可能です：

- **ディスクに書き込まないオンザフライ・ラスタを作成**
  - これにチェックを入れない場合、出力結果は通常の新しいファイルとしてディスク上に保存されます。 **出力レイヤ** のパスと、 **出力形式** を指定する必要があります。

  - チェックを入れた場合、仮想のラスタレイヤ、つまり URI によって定義され、そのピクセルがオンザフライで計算されるラスタレイヤが作成されます。これはディスク上の新しいファイルではありません。仮想レイヤは計算に使用したラスタに関連付いているため、計算に使用したラスタを削除したり移動したりすると、仮想レイヤが壊れてしまいます。 **レイヤ名** は指定することができますが、指定しない場合は計算式がレイヤ名として使用されます。仮想レイヤをプロジェクトから取り除くと削除されてしまいますが、レイヤの **エクスポート　名前を付けて保存**... コンテキストメニューを使用して、ファイルの形式でこのレイヤを永続化させることができます。

- 計算の **空間範囲** は、入力ラスタレイヤの領域に基づいて定義できますが、カスタムの X,Y 座標を入力することもできます。
- レイヤの **解像度** は、カラム数と行数で設定します。入力レイヤの解像度と異なる場合には、最近傍アルゴリズムを使用して値がリサンプリングされます。
- **結果をプロジェクトに追加する** のチェックボックスにチェックを入れると、結果のレイヤを自動的に凡例に追加し、レイヤを表示できます。仮想ラスタの場合にはデフォルトでチェックが入ります。

演算子 セクションには利用可能なすべての演算子があります。演算子をラスタ計算機の式ボックスに追加するには、それに応じたボタンをクリックします。算術計算（ `+` 、 `-` 、 `*` など）や三角関数（ `sin` 、 `cos` 、 `tan` など）が利用できます。条件式（ `=` 、 `!=` 、 `<` 、 `>=` など）は、偽の場合は 0 、真の場合は 1 を返すので、他の演算子や関数と共に使うことができます。

ヒント: **ラスタ計算機** のアルゴリズムについても参照してください。

例

標高値をメートルからフィートに変換する

メートル単位の標高ラスタからフィート単位のラスタを作成するには、メートルからフィートへの換算係数 3.28 を掛けます。式は次のとおりです：

```
"elevation@1" * 3.28
```

マスクを使用する

例えば、標高 0 メートル以上の部分だけに興味があるなど、ラスタの一部をマスクしたい場合には、マスクの作成とその結果のラスタへの適用が以下の式を使用して一度に行えます。

```
("elevation@1" >= 0) * "elevation@1"
```

これはつまり、標高値が 0 以上の各セルは条件式の評価が 1 になり、元の値に 1 を掛けるので値が維持されます。そうでない場合には条件式の評価が 0 になり、ラスタ値が 0 になります。これにより、標高値に応じたマスクが作成されます。

ラスタのクラス分類

ラスタを例えば 2 つの標高クラスに分類したい場合には、次の式を使用すると、1 つのステップで 2 種類の値 1 と 2 を持つラスタを作成することができます。

```
("elevation@1" < 50) * 1 + ("elevation@1" >= 50) * 2
```

これはつまり、標高値が 50 未満の各セルには値 1 が設定され、50 以上の各セルには値 2 が設定されます。

IF 演算子を使用することもできます。

```
if ( elevation@1 < 50 , 1 , 2 )
```

17.2.2 ラスタを揃える

このツールは複数のラスタを入力として、それらの位置を完全に合わせることが可能です。これはつまり、

- 同じ CRS に再投影し、
- 同じセルサイズや同じグリッドオフセットにリサンプリングし、
- 関心のある領域で切り抜きを行い、
- 必要に応じて値を再スケールします。

ラスタはすべて、別のファイルに保存されます。

まず、 **ラスタ** **ラスタを揃える**... からツールを開き、 新しいラスタを追加 ボタンをクリックして、QGIS の既存のラスタを 1 つ選択します。揃えたあとのラスタを保存する出力ファイルやリサンプリング方法を選択し、ツールで必要ならば **セルサイズに応じてスケール** にチェックを入れます。リサンプリング方法は以下が選択できます（ 図 17.22 参照）:

- **最近傍（Nearest Neighbor）**
- **バイリニア（2x2）**
- **キュービック（4x4 ）** ：三次の畳込み内挿近似
- **キュービック B スプライン（4x4 ）** ：キュービック B スプライン近似
- **ランチョス（6x6）** ：ランチョス窓による sinc 補間
- **平均（Average）** ：NODATA でない寄与ピクセルすべての平均を計算
- **モード** ：すべてのサンプリングポイントの中で最も頻繁に出現する値を選択
- **最大値** 、 **最小値** 、 **中央値** 、 **第１四分位（Q1）** 、 **第３四分位（Q3）** ：それぞれ NODATA でない寄与ピクセルすべての最大値 、最小値、中央値、第 1 四分位値、第 3 四分位値を選択

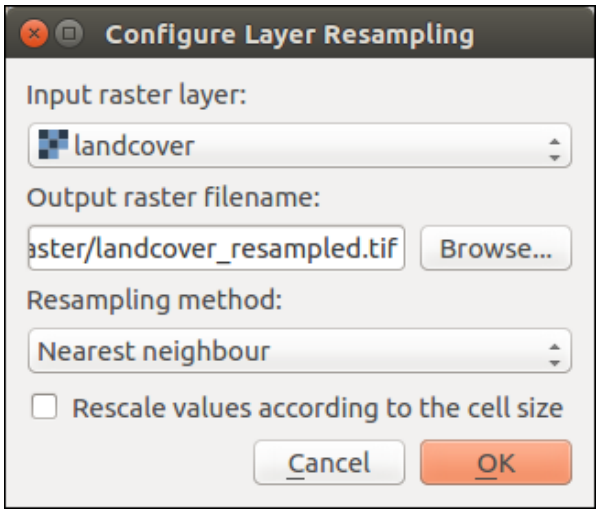


図 17.22: ラスタのリサンプリングオプションの選択

ラスタを揃える のメインダイアログでは、ファイル設定の編集 を行ったり、ラスタレイヤのリストから 既存のファイルを削除 することができます。また、以下の複数のオプションを選択することもできます (図 17.23 参照)。

- **スナップで参照するレイヤ** の選択
- 新しい **座標参照系（*CRS*）** への変換
- 異なる **セルサイズ** の設定
- 異なる **グリッドオフセット** の設定
- **切り抜く範囲** ：ユーザ定義の範囲やレイヤ領域、マップキャンバス領域を境界とすることができます
- **出力サイズ**
- **揃えられたラスタをキャンバスに追加する**

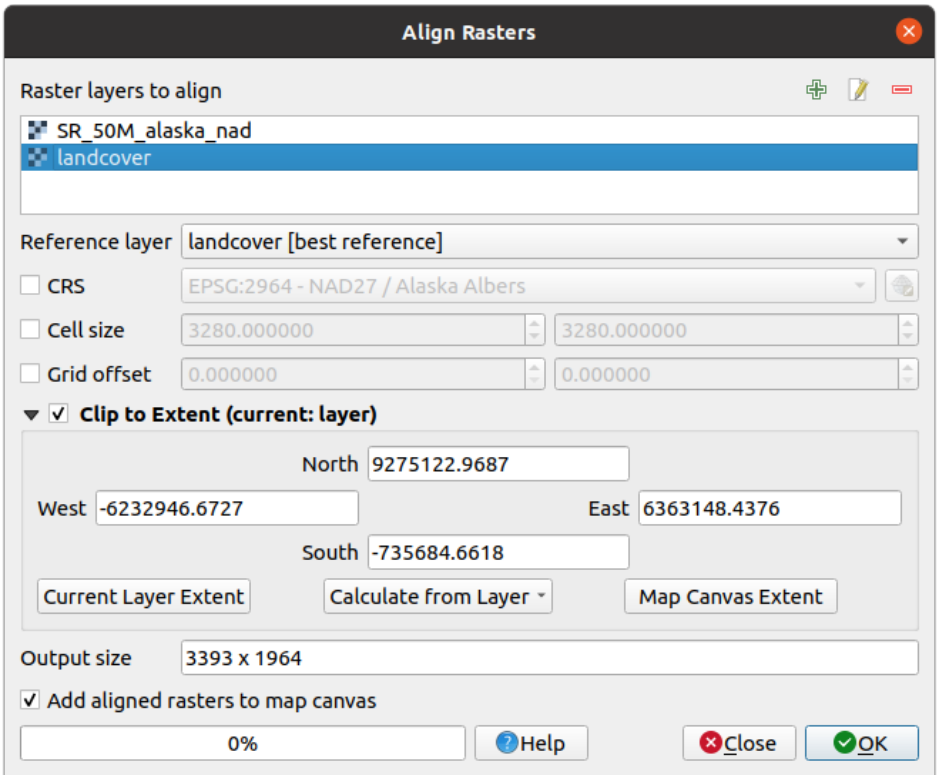


図 17.23: ラスタを揃える

17.3 ジオリファレンサ

ジオリファレンサは、レイヤのワールドファイルを生成するためのツールです。新しい GeoTiff を作成したり、既存の画像にワールドファイルを追加することで、ラスタやベクタを地理的または投影座標系に参照付けることができます。レイヤをジオリファレンスする基本的な方法は、レイヤ上で正確に座標を決定できる点を見つけることです。

機能

アイコン	目的	アイコン	目的
	ラスタを開く		ベクタを開く
	ジオリファレンスを開始		
	GDAL スクリプトを生成		GCP を読み込み
	名前を付けて保存		変換を設定
	点を追加		点を削除
	GCP を移動		パン
	拡大		縮小
	レイヤの全領域にズーム		前の領域へズーム
	次の表示領域にズーム		ジオリファレンサを QGIS にリンクする
	QGIS をジオリファレンサにリンクする		ヒストグラムをデータセット全域の値で引きのばす
	ヒストグラムを表示領域の値で引きのばす		

表：ジオリファレンサのツール

17.3.1 通常の手順

画像上の選択された点に対応する X、Y 座標（度分秒（dd mm ss.ss）や度（dd.dd)、あるいは投影された座標（mmmm.mm)）の設定手順には、 2 つの選択肢があります：

- ラスタ自体が、画像上に座標値のある十字が「描かれて」いる場合があります。この場合には、その座標値を手動で入力します。
- すでにジオリファレンスされたレイヤを使用する方法もあります。このレイヤはベクタデータでもラスタデータでもよく、ジオリファレンスしたい画像内にある同じオブジェクト/地物が含まれており、画像に使用したい座標系を持っているものです。この場合には、QGIS のマップキャンバスにロードされた参照データセット上をクリックして、座標を入力できます。

画像をジオリファレンスする通常の手順では、ラスタ上で複数の点を選択し、それらの座標を指定して、関連する変換タイプを選択します。入力パラメータと入力データに基づいて、ジオリファレンサはワールドファイルのパラメータを計算します。より多くの座標を指定するほど、より良い結果が得られるでしょう。

まず、QGIS を起動し、QGIS のメニューバーに表示されている **レイヤ** **ジオリファレンサ** をクリックします。図 17.24 に示すように、ジオリファレンサダイアログが表示されます。

以下の例では、SDGS によるサウスダコタ州の地形図を使用します。これは後で GRASS の `spearfish60` のデータと一緒に可視化します。地形図は以下のリンクからダウンロードできます：https://grass.osgeo.org/sampledata/spearfish_toposheet.tar.gz

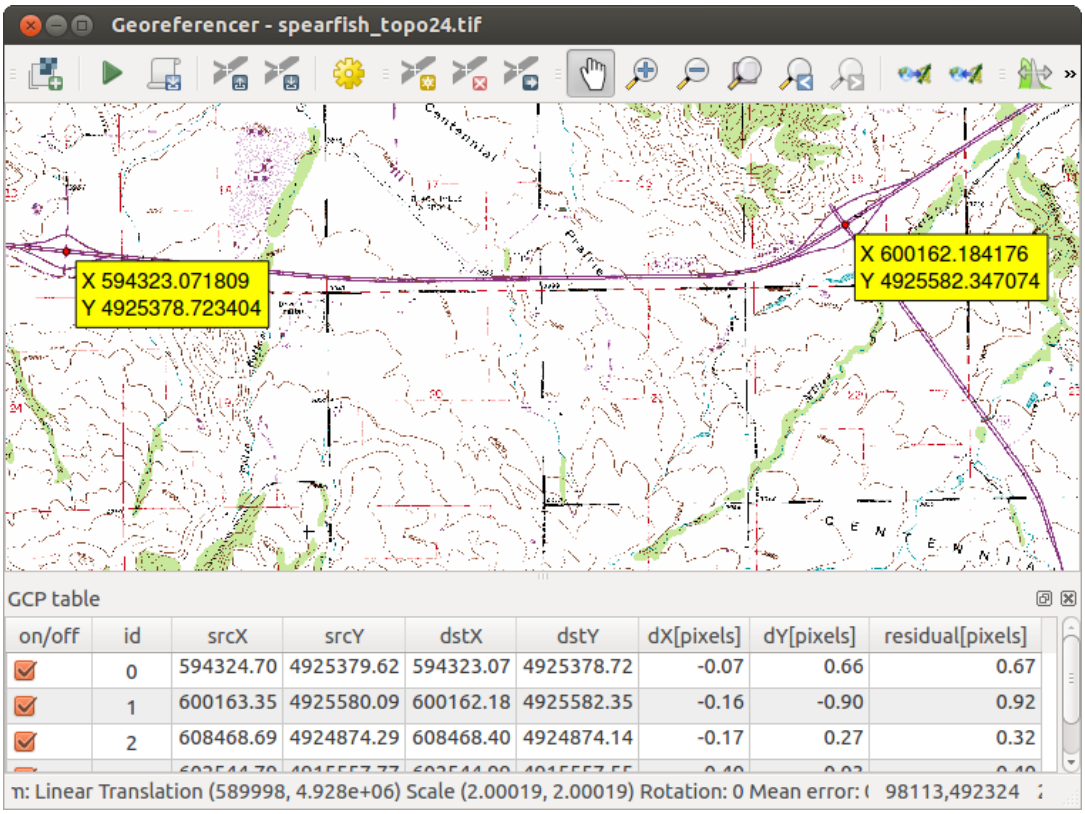


図 17.24: ジオリファレンサダイアログ

グラウンドコントロールポイント（GCP）の入力

1. ジオリファレンスされていないラスタのジオリファレンスを開始するには、 ボタンを使用してそのラスタをロードする必要があります。ラスタはダイアログのメインの作業領域に表示されます。ラスタがロードされると、基準点の入力を開始できます。

2. 点を追加 ボタンを使用して、メインの作業エリアにポイントを追加し、それらの座標を入力します（ 図 17.25 を参照）。この手順には 3 つの方法があります：

   - ラスタ画像内の点をクリックして、その点の CRS とともに X 座標と Y 座標を手動で入力する。
   - ラスタ画像内の点をクリックし、 地図キャンバスから ボタンを押して、既に QGIS のマップキャンバスに読み込まれたジオリファレンス済みの地図を使用して、X 座標と Y 座標を追加する。CRS は自動的に設定されます。

3. 点の入力を繰り返します。最低 4 点の指定が必要ですが、より多くの座標を入力すれば、より良い結果が得られます。また、関連する GCP ポイントの組を適切な場所に置くために、作業領域をズームしたりパンしたりするツールもあります。

4. ツールを使用すると、GCP を修正する必要がある場合に、マップキャンバスとジオリファレンスウィンドウの両方で GCP を移動させることができます。

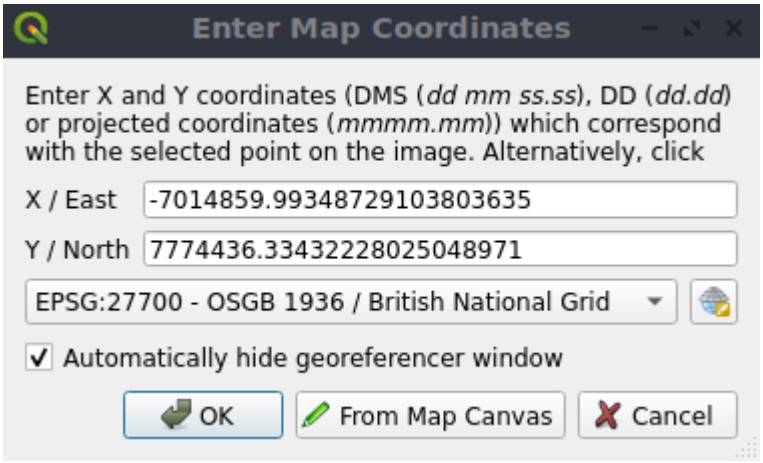


図 17.25: ラスタ画像へポイントを追加

地図に追加した点は通常、ラスタ画像と同じ場所の別個のテキストファイル（`[ファイル名].points`）に保存されます。これにより、後日ジオリファレンサを再度開いて、新しい点を追加したり既存の点を削除して結果を最適化することができます。このポイントファイルには、次のような形式の値が含まれています：`mapX, mapY, pixelX, pixelY`。 GCP を読み込み ボタンと 名前を付けて保存 ボタンを使用して、ファイルの管理ができます。

変換の設定の定義

ラスタ画像に GCP を追加したら、ジオレファレンス処理のための変換の設定を定義する必要があります。

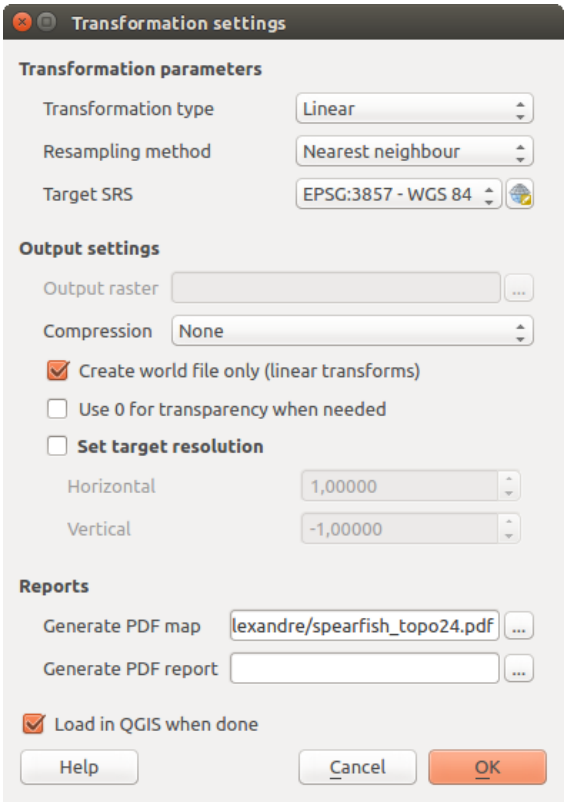


図 17.26: ジオリファレンサの変換設定の定義

利用可能な変換アルゴリズム

入力データのタイプや品質、最終結果に入ることを許容する幾何学的な歪みの性質や量、グラウンドコントロールポイント（GCP）の数に応じて、多数の変換アルゴリズムが利用可能です。

現在、以下の **変換タイプ** が利用できます：

- **線形** アルゴリズムはワールドファイルの作成に使用され、他のアルゴリズムとは異なり、実際にはラスタピクセルの変換は行われません。これは画像の位置合わせ（平行移動）と一様な拡大縮小は行いますが、回転やその他の変換は行いません。画像が高品質のラスタマップで、CRS は既知だが地理参照情報が不足している場合にはこのアルゴリズムが最適です。最低でも 2 つの GCP が必要です。
- **ヘルマート** 変換では、回転も可能です。ラスタが高品質の大縮尺地図やオルソ画像で、CRS のグリッド方位と一致していない場合には、この変換が特に有用です。最低でも 2 つの GCP が必要です。
- **多項式 1** アルゴリズムはより一般的なアフィン変換で、一様せん断変形も行います。変形後は直線は直線のまま（つまり一直線上の点は一直線上のまま）、平行な線は平行なままです。このアルゴリズムは、異なる方向に異なるグラウンドピクセルサイズでプロットされている（またはデータが取得されている）可能性のあるデータカートグラムのジオリファレンスに特に有用です。 最低でも 3 つの GCP が必要です。
- **多項式 2-3** アルゴリズムは、単なるアフィン変換ではなく、より一般的な 2 次または 3 次の多項式を使用します。このアルゴリズムでは、例えばエッジが曲がっている撮影地図などにある、曲率やその他の系統的なゆがみをを考慮できます。少なくとも 6 つ（多項式 3 は 10 個）の GCP が必要です。角度は保存されず、局所的な縮尺は画像全体で均一にはなりません。特に、直線が曲がってしまった

り、データに合わせた多項式を大きく外挿させることによって、エッジや GCP から離れた場所で大きなゆがみを生じる可能性があります。

- **投影変換** アルゴリズムは多項式 1 を別の方法で一般化したもので、画像とマップキャンバスという 2 つの平行でない平面間の中心射影を表現する変換ができます。直線は変換後も直線に保たれますが、平行性は保存されず、画像全体のスケールは視点の変化に一致するように変化します。この変換タイプは、高品質の地図の（垂直スキャンではなく）傾いた写真や、斜め撮影の航空写真をジオリファレンスする際に最も有用です。最低でも 4 つの GCP が必要です。
- **薄板スプライン**（Thin Plate Spline）アルゴリズムは、複数のローカルな多項式を用いてラスタを「ラバーシート化」し、全体の表面曲率を最小化しながら指定された GCP に一致させます。出力結果では、GCP から離れた領域は GCP のマッチングに合わせて移動することはありますが、局所的な変形は最小限に抑えられます。薄板スプラインは、破損した地図や変形した地図、微妙に不正確な地図、あるいはオルソ化されていない空中写真などのジオリファレンスに最も有用です。またこれは、投影タイプや投影パラメータが不明ではあるが、規則的なグリッドやアドホックな GCP の密な組み合わせを参照地図レイヤとマッチさせることのできる地図に対して、近似的なジオリファレンスや暗黙的な再投影を行う場合にも有用です。技術的には最低でも 10 個の GCP が必要ですが、良好な結果を得るためには通常、より多くの GCP が必要です。

薄板スプライン以外のアルゴリズムはすべて、必要最低限の GCP よりも多くの GCP が指定された場合、全体の残差誤差が最小になるようにパラメータをフィッティングします。これは、座標入力時の誤差、すなわち、ポインタのクリックや入力座標のわずかな不正確さや、その他の小さな局所的な画像変形による影響を最小限に抑えるのに役立ちます。補正するための GCP が他にない場合、このような誤差や変形は、ジオリファレンスされた画像のエッジ付近で特に大きな歪みとなる可能性があります。ただし、GCP を必要最低数よりも多く指定した場合には、出力画像の GCP は近似的にしか一致しません。一方で、薄板スプラインは指定した GCP すべてに正確にマッチしますが、座標入力時誤差がある GCP の周辺では大きな変形が生じる可能性があります。

リサンプリング方法の定義

どのようなリサンプリング方法を選択するかは、入力データと作業の最終目的によって異なります。（線形、ヘルマート、多項式 1 以外の変換を使用することによる不均一な幾何学的スケーリングによる影響以外で）ラスタの統計情報を変更したくない場合には、「最近傍（Nearest neighbour）」を選択するとよいでしょう。一方で、例えば「キュービック（Cubic）」リサンプリングはほとんどの場合、より視覚的に滑らかな結果が得られます。

5 つの異なるリサンプリング方法から選択できます：

1. 最近傍（Nearest neighbour）
2. 線形
3. キュービック（Cubic）
4. キュービック・スプライン（Cubic Spline）
5. ランチョス法（Lanczos）

変換の設定の定義

ジオリファレンスされた出力ラスタの作成のために、さまざまなオプションを指定する必要があります。

- **ワールドファイルの作成のみ（リニア変換）**のチェックボックスは、線形変換タイプを指定した場合にのみ利用可能です。これは、実際にはラスタ画像は変換されないことを意味しています。チェックを入れた場合、新しいワールドファイルが作成されるのみのため、 **出力ラスタ** フィールドは無効になります。
- これ以外の変換タイプに対しては、 **出力ラスタ** を指定する必要があります。デフォルトでは、新規ファイル（[ファイル名]_modified）をオリジナルのラスタ画像ファイルと同じフォルダに作成します。
- 続いて、ジオリファレンスされたラスタの **変換先** *SRS*（空間参照系）を指定する必要があります（ **投影法の利用方法** 参照）。
- 必要ならば **PDF マップを作成** や **PDF レポートを作成** できます。 このレポートには、使用した変換パラメータ、残差の画像、すべての GCP とその RMS エラーのリストといった情報が含まれています。
- さらに、 **解像度を設定** チェックボックスにチェックを入れると、出力ラスタのピクセル解像度を指定できます。デフォルトの水平解像度と垂直解像度は 1 です。
- **必要に応じて透明に *0* を使用** にチェックを入れると、値 0 のピクセルは透明で表示されるようになります。サウスダコタ地形図の例では、余白領域がすべて透明になります。
- 最後に、 **完了時に *QGIS* にロードする** にチェックを入れると、変換が完了したときに出力ラスタを QGIS のマップキャンバスに自動的に読み込みます。

ラスタプロパティの表示と調整

設定 メニューの **ラスタプロパティ** オプションをクリックすると、ジオリファレンスしようとしているラスタファイルの **レイヤプロパティ** ダイアログが開きます。

ジオリファレンサの構成

- GCP の座標や ID を表示するかどうかを定義できます。
- 残差の単位として、ピクセルまたは地図上の単位を選択できます。
- PDF レポートについて左右のマージンを指定したり、PDF 地図について用紙サイズを指定できます。
- 最後に、 **ジオリファレンサウィンドウを結合して表示する** を有効化できます。

変換の実行

すべての GCP を設定しすべての変換設定が定義されたら、  ジオリファレンスを開始 ボタンを押すと、新しいジオリファレンスされたラスタが作成されます。

第18章　メッシュデータの操作

18.1 メッシュとは？

メッシュとは非構造格子のことで、通常、時間的な要素やその他の要素を持っています。空間成分には、2D または 3D 空間の頂点、辺、面の集合が含まれます。

- **頂点** - （レイヤの座標参照系の）XY(Z) 成分を持つ点です。
- **辺** - 頂点の組を接続します。
- **面** - 面は閉じた形状を形成する辺の集合です。通常は三角形または四辺形（quad）で、頂点数が多い多角形はめったにありません。

上記の要素に基づいて、メッシュレイヤはさまざまなタイプの構造を持つことができます：

- 1D メッシュ：頂点と辺で構成されます。辺は 2 つの頂点を接続し、その上にデータ（スカラまたはベクタ）を割り当てることができます。 1D メッシュネットワークは、たとえば、都市の排水システムのモデリングに使用できます。
- 2D メッシュ：三角形の面や、構造格子または非構造格子の四角形の面で構成されます。
- 3D レイヤードメッシュ：複数の積み重ねられた 2D 非構造メッシュで構成され、それぞれが垂直座標によって垂直方向（レベル）に押し出されたものです。頂点と面は、各垂直レベルで同じトポロジを持っています。メッシュ定義（垂直レベルの押し出し）は、一般的には時間とともに変化する可能性があります。データは通常、ボリュームの中心か、またはパラメトリック関数によって定義されます。

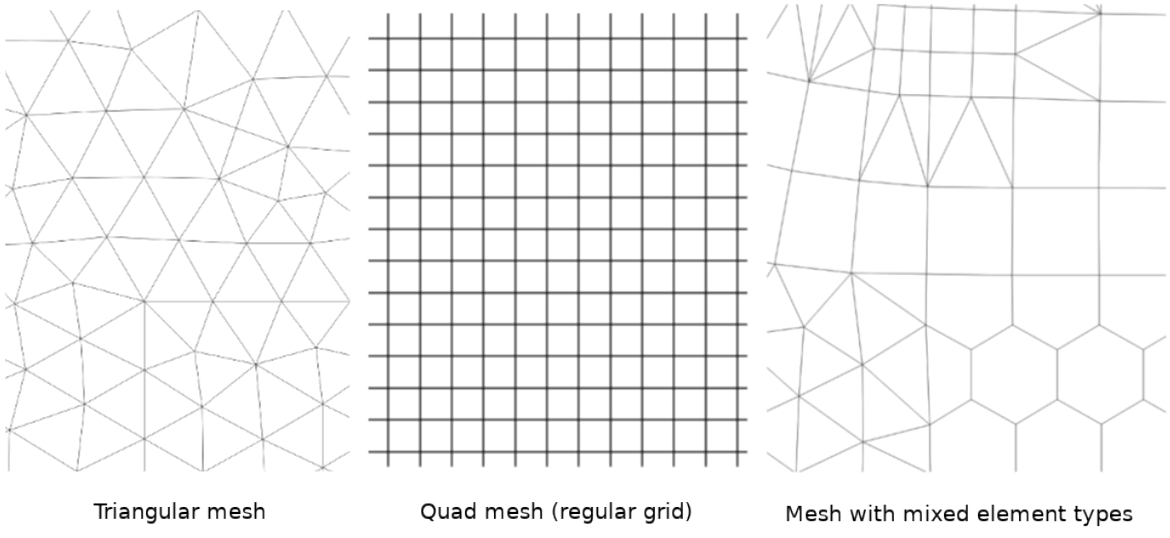


図 18.1: さまざまなメッシュタイプ

メッシュは空間構造に関する情報を提供します。さらに、メッシュはすべての頂点に値を割り当てるデータセット（グループ）を持つことができます。たとえば、次の図に示すように頂点が番号付けされた三角メッシュがあるとします：

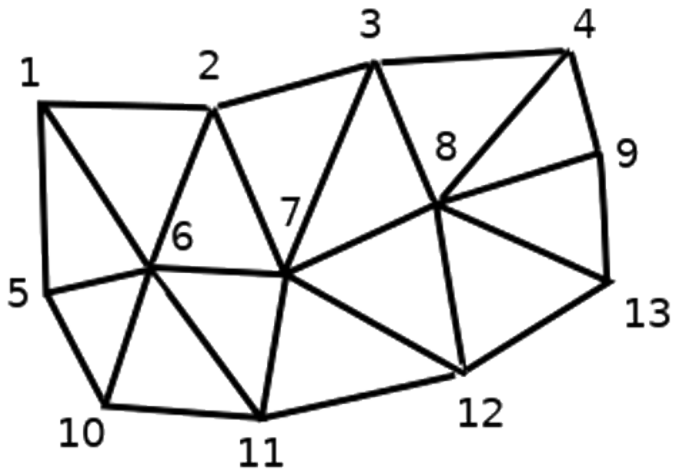


図 18.2: 頂点に番号が付けられた三角形グリッド

各頂点は異なるデータセット（通常は複数の量）を格納でき、それらのデータセットは時間的な次元を持つこともできます。したがって、1 つのファイルに複数のデータセットを含めることができます。

以下のテーブルは、メッシュデータセットに格納できる情報についての概念を示しています。テーブルの列はメッシュの頂点のインデックスを表し、各行は 1 つのデータセットを表します。データセットはさまざまなデータ型を持つことができます。この例の場合には、ある特定の時間（t1、t2、t3）における地上 10m の風速がテーブルに格納されています。

同様に、メッシュデータセットは各頂点のベクトル値も格納できます。たとえば、与えられたタイムスタンプでの風向ベクトルは、

表 18.1: メッシュデータセットの例

地上 10m 風速	1	2	3	...
time=t1 における地上 10m 風速	17251	24918	32858	...
time=t2 における地上 10m 風速	19168	23001	36418	...
time=t3 における地上 10m 風速	21085	30668	17251	...
...	...	...	...	...
time=t1 における地上 10m 風向	[20,2]	[20,3]	[20,4.5]	...
time=t2 における地上 10m 風向	[21,3]	[21,4]	[21,5.5]	...
time=t3 における地上 10m 風向	[22,4]	[22,5]	[22,6.5]	...
...	...	...	...	...

色を値に割り当てることでデータを可視化することができ（**単バンド疑似カラー** のラスタレンダリングと同様です）、メッシュトポロジに従って頂点間のデータが補間されます。いくつかの量は単純なスカラー値ではなく、2D ベクトル（例えば風向）であるのはよくあることです。そのような量については、方向を示す矢印を表示することが望ましいでしょう。

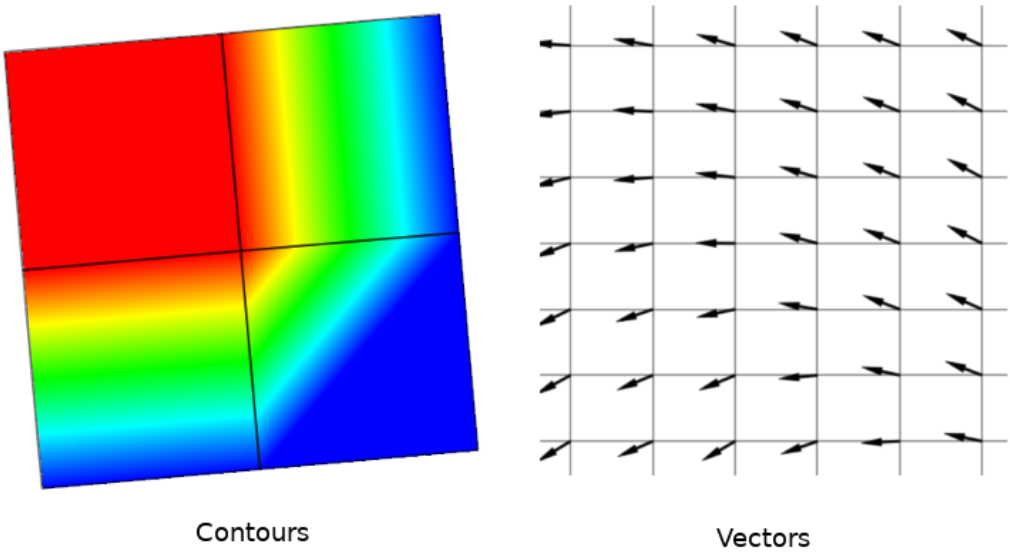


図 18.3: メッシュデータの可視化事例

18.2 サポートする形式

QGIS は MDAL ドライバ を使用してメッシュデータにアクセスし、 さまざまなフォーマット をネイティブにサポートしています。QGIS でメッシュレイヤを編集できるかどうかは、フォーマットとメッシュ構造タイプによります。

メッシュデータセットを QGIS に読み込むには、 **データソースマネージャ** ダイアログで **メッシュ** タブを使用します。詳細は、 **メッシュレイヤを読み込む** を参照してください。

18.3 メッシュデータセットのプロパティ

メッシュレイヤの **レイヤプロパティ** ダイアログは、レイヤのデータセットグループとそのレンダリング（アクティブなデータセットグループ、シンボロジ、2D および 3D レンダリング）を管理するための一般的な設定を提供します。また、レイヤに関する情報も提供します。

レイヤプロパティ ダイアログにアクセスするには：

- **レイヤ** パネル内でレイヤをダブルクリックするか、レイヤを右クリックして、ポップアップメニューから **プロパティ**... を選択する
- レイヤが選択された状態で、 **レイヤ** **レイヤのプロパティ**... メニューを選ぶ

メッシュレイヤの **レイヤプロパティ** ダイアログには、以下のセクションがあります：

表 18.2: メッシュレイヤプロパティのタブ

情報	ソース	シンボロジ [1]
3D ビュー [1]	時系列	標高
レンダリング	メタデータ	

[1] **レイヤスタイルパネル** からも利用可能です

注釈: メッシュレイヤのプロパティの大半は、ダイアログの下部にある **スタイル** メニューを使用して `.qml` ファイルに保存したり、読み込んだりすることができます。詳細については、 **カスタムスタイルを管理する** を参照してください。

18.3.1 情報プロパティ

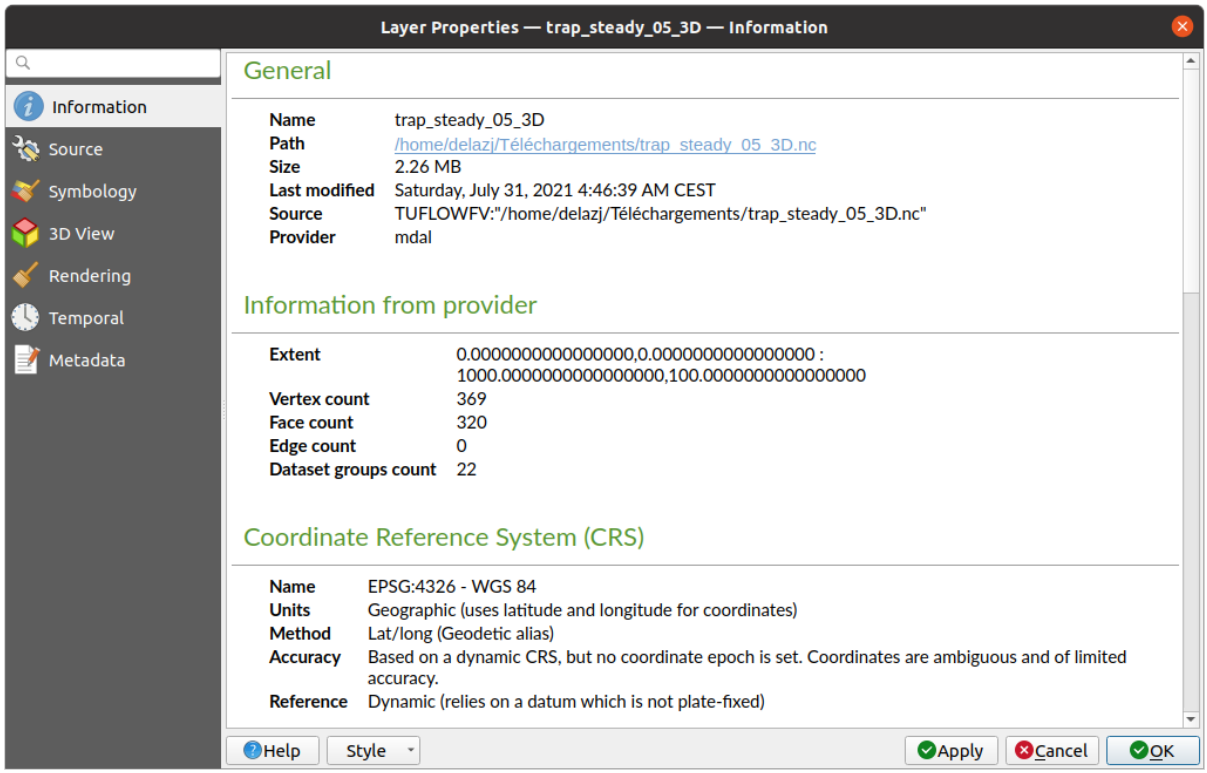


図 18.4: メッシュレイヤの情報プロパティ

情報 タブは読み取り専用で、現在のレイヤの要約された情報やメタデータをさっと掴むことができる興味深い場所です。提供される情報には、以下のものがあります：

- 一般情報：プロジェクト内での名前、ソースへのパス、付随的なファイルのリスト、最終更新時刻、ファイルの大きさ、使用しているプロバイダ
- プロバイダからの情報：領域、頂点の数、面の数、エッジ数、データセット数

- 空間参照システム（CRS）：CRS の名前、単位、投影法、精度、参照（静的か動的か）
- 入力された **メタデータ** からの情報：アクセス、領域、リンク、連絡先、履歴など

18.3.2 ソースプロパティ

ソース タブは、選択されたメッシュに関する以下のような基本的な情報を表示します：

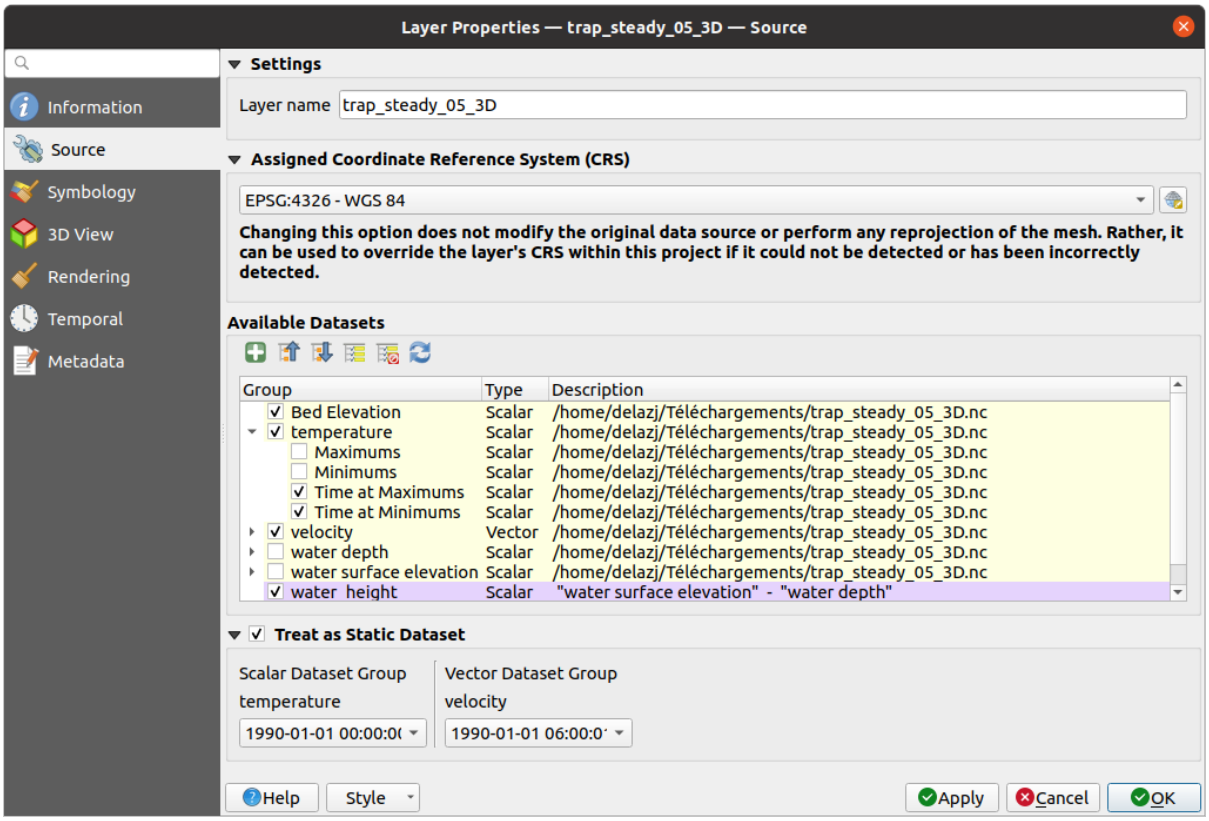


図 18.5: メッシュレイヤのソースプロパティ

- **レイヤ** パネルで表示されるレイヤ名
- 座標参照系の設定：レイヤに **設定された** *CRS* を表示します。最近使用した CRS をドロップダウンリストから選ぶか、 *CRS* **の選択** ボタン（ **座標参照系セレクタ** 参照）をクリックすることで、レイヤの CRS を変更できます。レイヤの CRS が間違っている場合か、CRS が何も設定されていない場合にのみ、この操作を行ってください。
- **利用可能なデータセット** のフレームには、メッシュレイヤ内のすべてのデータセットグループ（とサブグループ）が、その種類や説明とともにツリービュー形式でリストされています。通常のデータセット（すなわちファイル内に保存されたデータ）と、仮想データセット（ **オンザフライで計算されたもの** ）の両方が表示されます。
  - **メッシュに追加データセットを割り当てる** ボタンを使用すると、現在のメッシュレイヤにさらにデータグループを追加します。
  - **すべて折りたたむ** と **すべて展開する** は、グループが埋め込まれている場合にデータセットのツリーを折りたたみ・展開します

  - 一部のデータセットにしか興味がない場合、他のデータセットのチェックを外し、プロジェクト内で利用不可にできます。
  - グループ名をダブルクリックすると、データセットの名前を変更できます。
  - **デフォルトにリセット** ：グループのすべてにチェックを入れ、データ名をプロバイダの元の名前に戻します。
  - 仮想データセットグループ上で右クリックすると、以下の操作が可能です：
    * プロジェクトから **データセットグループを削除**
    * **データセットグループを別名で保存**... ：ディスク上のファイルにサポートする任意の形式で保存します。新しいファイルはプロジェクト内で現在のメッシュレイヤに割り当てられます。

- *Treat as static dataset* グループにチェックを入れると、メッシュレイヤのレンダリング時に **マップの時系列ナビゲーション** プロパティを無視できます。アクティブなデータセットグループ（ **シンボロジ** **データセット** タブで選択されたグループ）のそれぞれについて、次の設定ができます：
  - *none* に設定：データセットグループは完全に表示されません
  - **データセットを表示** ：例えば、時間とは無関係な「地盤高」データセットに使用します
  - 特定の日時を抽出：指定した時刻にマッチするデータセットをレンダリングし、マップの時系列ナビゲーション中はこれで固定します

18.3.3 シンボロジプロパティ

シンボロジ ボタンをクリックすると、ダイアログを有効化します。シンボロジのプロパティは、複数のタブに分かれています：

- **データセット**
- **等高線**（*Contours*）
- **ベクタ**
- **レンダリング**
- **多層メッシュ平均化法**（*Stacked mesh averaging method*）

データセット


データセット タブはレイヤにどのデータセットを使用するかの制御や設定を行う中心的な場所です。以下のアイテムがあります：

- **グループ** ：メッシュデータセットで利用可能なグループと、以下のデータがあるかどうかを表示します：
  -  スカラデータセット
  -  ベクタデータセット：デフォルトでは、各ベクタデータセットは自動的に生成されたベクタの大きさを表すスカラデータセットを持ちます

  データセット名の横にあるアイコンをクリックすると、表示するデータのグループとタイプの選択ができます。
- **選択データセットグループのメタデータ** は、以下についての詳細を表示します：
  - メッシュ型：辺または面
  - データタイプ：頂点、辺、面またはボリューム
  - ベクトル型かどうか
  - メッシュレイヤのオリジナルの名前
  - 単位（あれば）
- 選択したデータセットに対する **混合モード** が利用可能です。

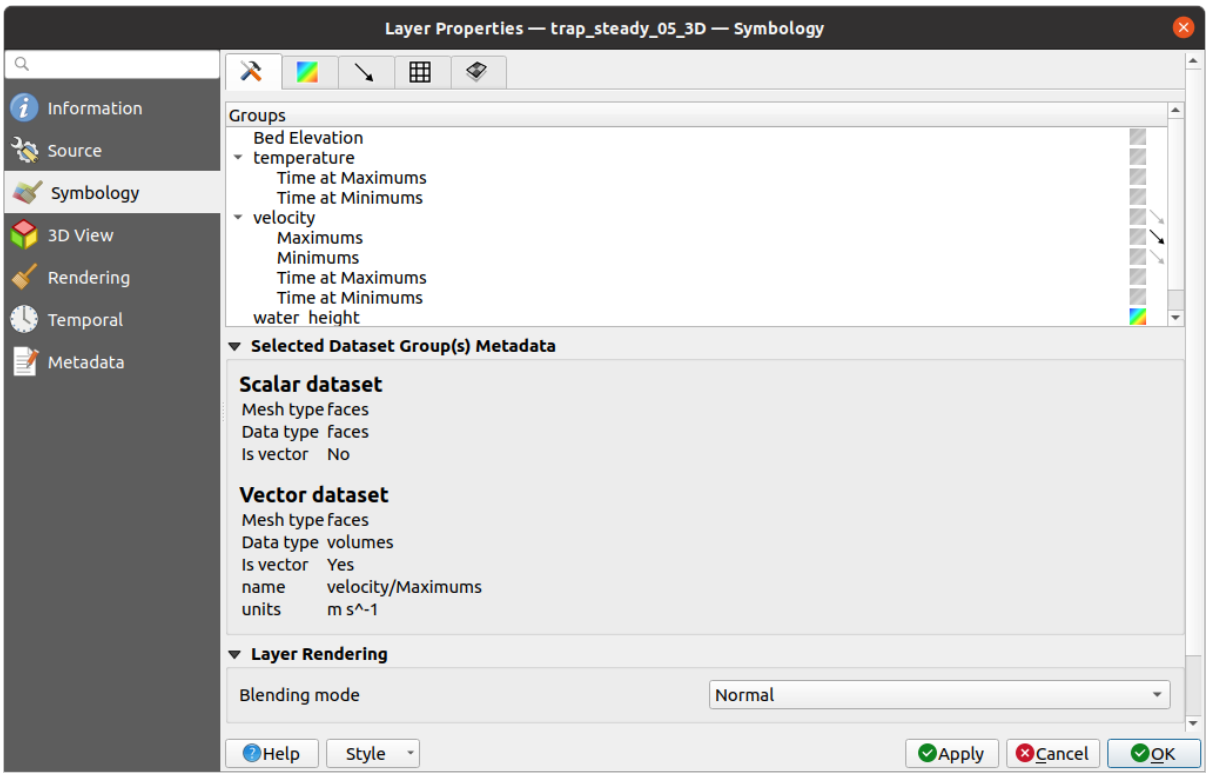


図 18.6: メッシュレイヤのデータセット

選択したベクトルやスカラーのグループには、次のタブを使用してシンボロジを適用できます。

等高線シンボロジ

注釈: 等高線（Contours）タブは、**データセット** タブでスカラーデータセットが選択されている場合にのみ、有効化することができます。

等高線（Contours）タブでは、以下の 図 18.7 に示すように、選択したグループに対するコンター表示の現在の可視化オプションの確認と変更ができます。

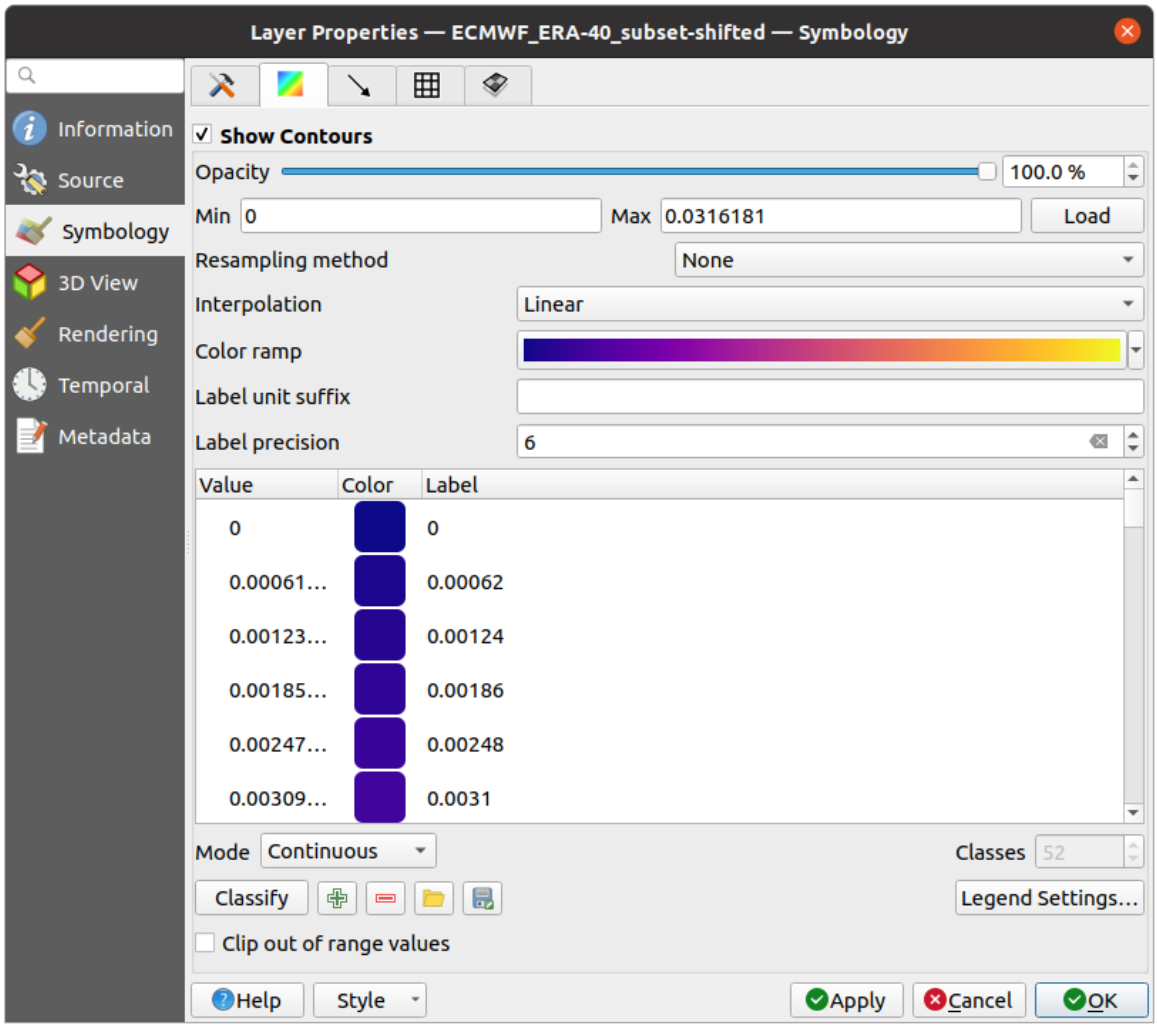


図 18.7: メッシュレイヤのコンタースタイルの設定

- 1D のメッシュに対しては、辺の **ストローク幅** の設定があります。ストローク幅はデータセット全体で一定のサイズとすることも、ジオメトリに沿って変化させる（詳細は **補間された線のレンダラー** を参照）こともできます。
- 2D メッシュタイプの場合は、スライダーやスピンボックスを使用して現在のグループの **不透明度** が設定できます。

- 現在のグループで表現したい値の範囲を入力します。 読み込み ボタンを使用して現在のグループの最小値と最大値を取得できますが、範囲の一部を除外したい場合には自身で値を入力してください。
- 2D / 3D メッシュの場合、 **リサンプリング方法** で **隣接平均** を使用すると、周囲の頂点の値を面に（あるいは周囲の面から頂点に）補間できます。データセットが頂点で定義されているか（あるいは面で定義されているか）に応じて、QGIS はこの設定を **なし** （面定義の場合は **隣接平均** ）にデフォルトで設定することで、頂点上の値を使用してデフォルトのレンダリングが滑らかになるようにします。
- **カラーランプシェーダ** のクラス分類を使用して、データセットをクラス分けできます。

ベクタシンボロジ

注釈: ベクタ タブは、 **データセット** タブでベクトルデータセットが選択されている場合にのみ、有効化することができます。

ベクタ タブでは、 図 18.8 に示すように、選択したグループに対するベクトル表示の現在の可視化オプションの確認と変更ができます。

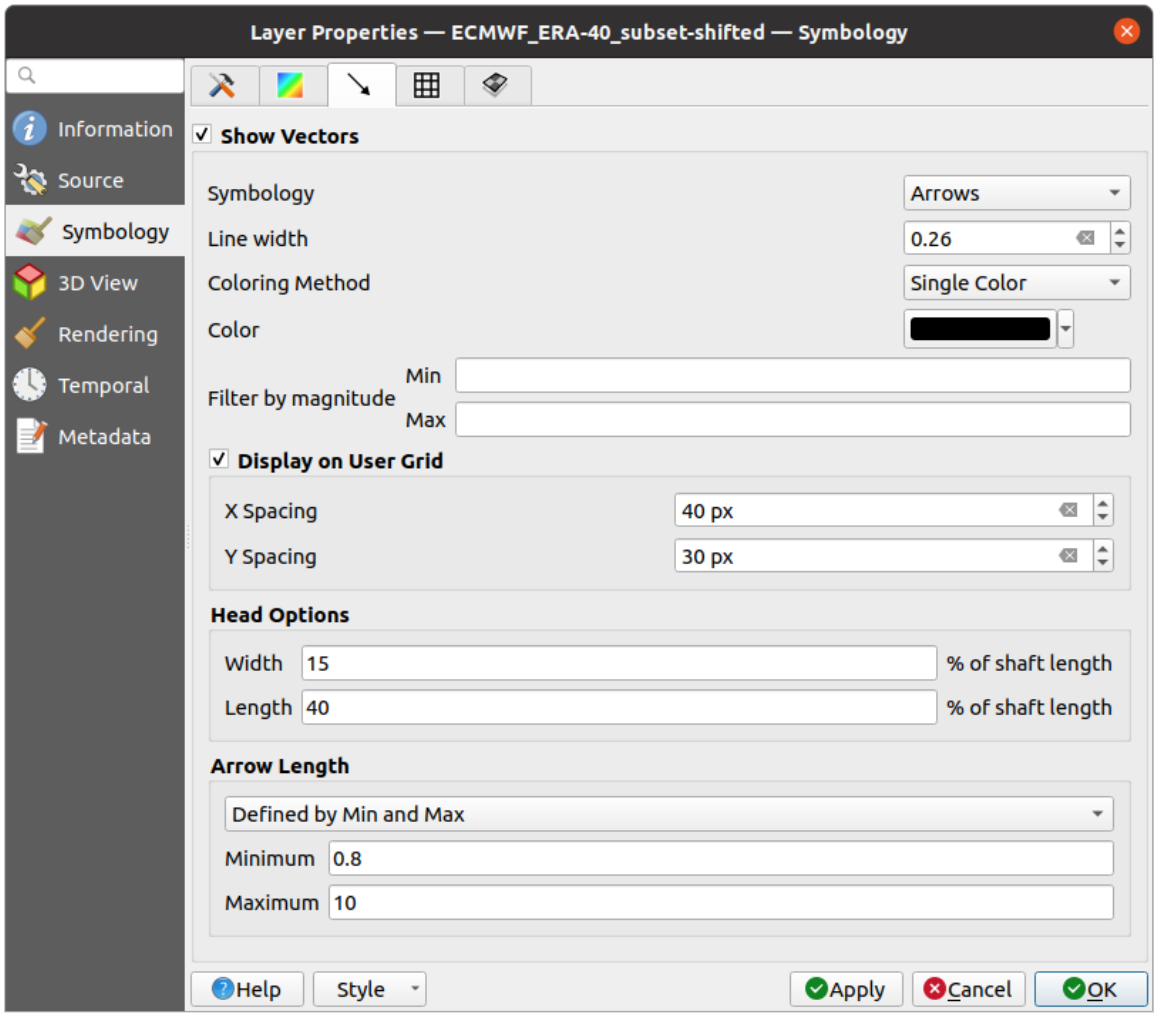


図 18.8: 矢印を使用したメッシュレイヤのベクトルスタイルの設定

メッシュのベクタデータセットは、さまざまな種類の **シンボロジ** を使用してスタイル設定ができます：

- **矢印** ：ベクトルは、生のデータセットでの定義位置（つまり、ノード上または要素の中心）と同じ位置、またはユーザー定義のグリッド上（したがって、均等に分布します）に矢印で表現されます。矢印の長さは生データで定義されている矢印の大きさに比例しますが、さまざまな方法でスケーリングできます。
- **流線** ：ベクトルは開始点から発生する流線で表現されます。流線の発生点はメッシュの頂点やユーザー定義のグリッド、あるいはランダムとすることができます。
- **トレース** ：流線のより良いアニメーションです。これは、水中にランダムに砂（トレーサー）を投げ入れ、それがどのように流れるかを見るような効果が得られます。

以下の表に示すように、利用可能なプロパティは選択したシンボロジによって異なります。

表 18.3: ベクタシンボロジのプロパティの有無と意味

Label	説明とプロパティ	矢印	流線	トレース
線幅	ベクトルを表現するシンボルの幅	☑	☑	☑
色付け方法	• **単一色** を全てのベクトルに適用する • **カラーランプシェーダ** を使用して、ベクトルの大きさに基づいた可変色とする	☑	☑	☑
大きさでフィルタリング	選択したデータセットのうち、大きさが **最小値** (*Min*) から **最大値** (*Max*) の範囲内にあるベクトルのみを表示する	☑	☑	
グリッド上に表示	*X* **方向の間隔** と *Y* **方向の間隔** によるユーザー定義のグリッド上にベクトルを配置する。ベクトルの大きさは近傍の値から内挿される	☑	☑	
ヘッドオプション	矢印のヘッドの **長さ** と **幅** を、矢印の軸の長さに対する百分率で指定する	☑		
矢印の長さ	• **最大・最小値で定義** :矢印の最小長さと最大長さを指定すると、QGIS は矢印の大きさをベクトルの大きさに基づいて補間する • **大きさにス**	☑		

レンダリング

レンダリング タブには、QGIS が提供するメッシュ構造の表示とカスタマイズの機能があります。表示のための **線幅** と **線の色** を設定できます：

- 1D メッシュの辺
- 2D メッシュの場合：
  - **ネイティブメッシュレンダリング** ：レイヤのオリジナルの面と辺を表示する
  - **三角メッシュレンダリング** ：辺を追加して、面を三角形として表示する

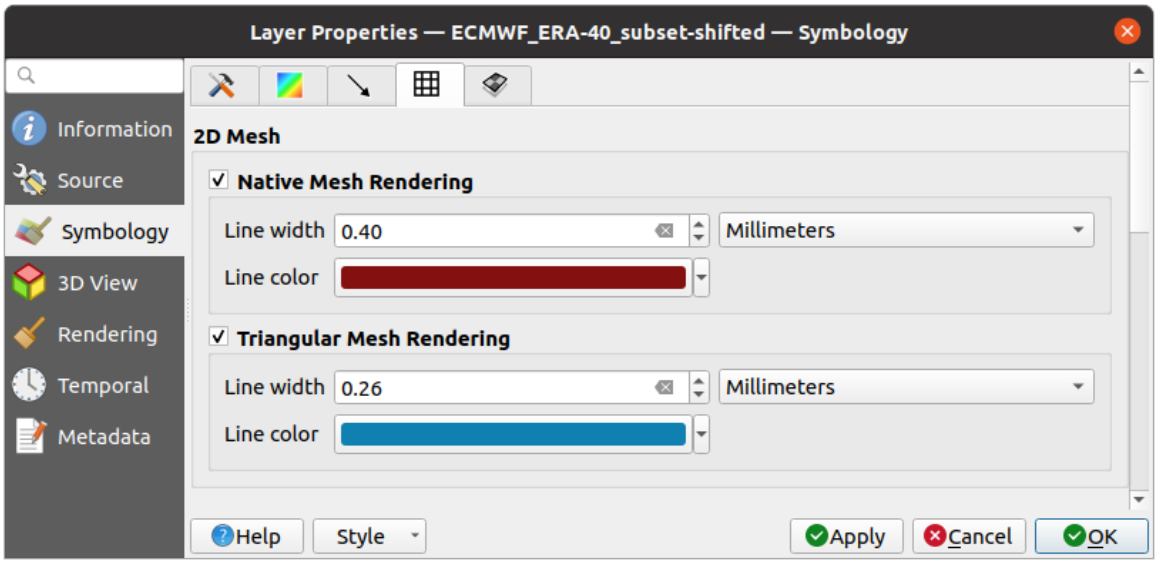


図 18.9: 2D メッシュのレンダリング

多層メッシュ平均化法

3D レイヤードメッシュは、2D 非構造メッシュを垂直座標によって垂直方向（ レベル ）に押し出したものが複数積み重なったもので構成されます。頂点と面は、各垂直レベルで同じトポロジーを持ちます。値は通常、ベースとなる 2D メッシュの上に規則的に積み重ねられたボリュームに保存されます。2D キャンバス上でこれを可視化するためには、ボリューム（3D）の値をメッシュレイヤで表示可能な面（2D）上の値に変換する必要があります。 多層メッシュ平均化法（Stacked mesh averaging method）は、これを処理するためのさまざまな平均化 / 補間方法を提供します。

2D のデータセットを導出するための方法と、それに対応するパラメータ（レベルインデックス、深度、標高など）を選択できます。各方法について、ダイアログに適用例が示されていますが、詳細については https://fvwiki.tuflow.com/index.php?title=Depth_Averaging_Results にあります。

18.3.4 3D ビュープロパティ

メッシュレイヤは頂点の Z 値に基づいて *3D* **マップビューの地形** としても使用できます。 *3D* **ビュー** プロパティタブから、メッシュレイヤのデータセットを同じ 3D ビューでレンダリングすることもできます。従って、頂点の鉛直成分はデータセット値（例えば水面の標高）と等しいように設定することができ、メッシュのテクスチャは他のデータセット値（例えば速度）をカラーランプシェーダでレンダリングするように設定できます。

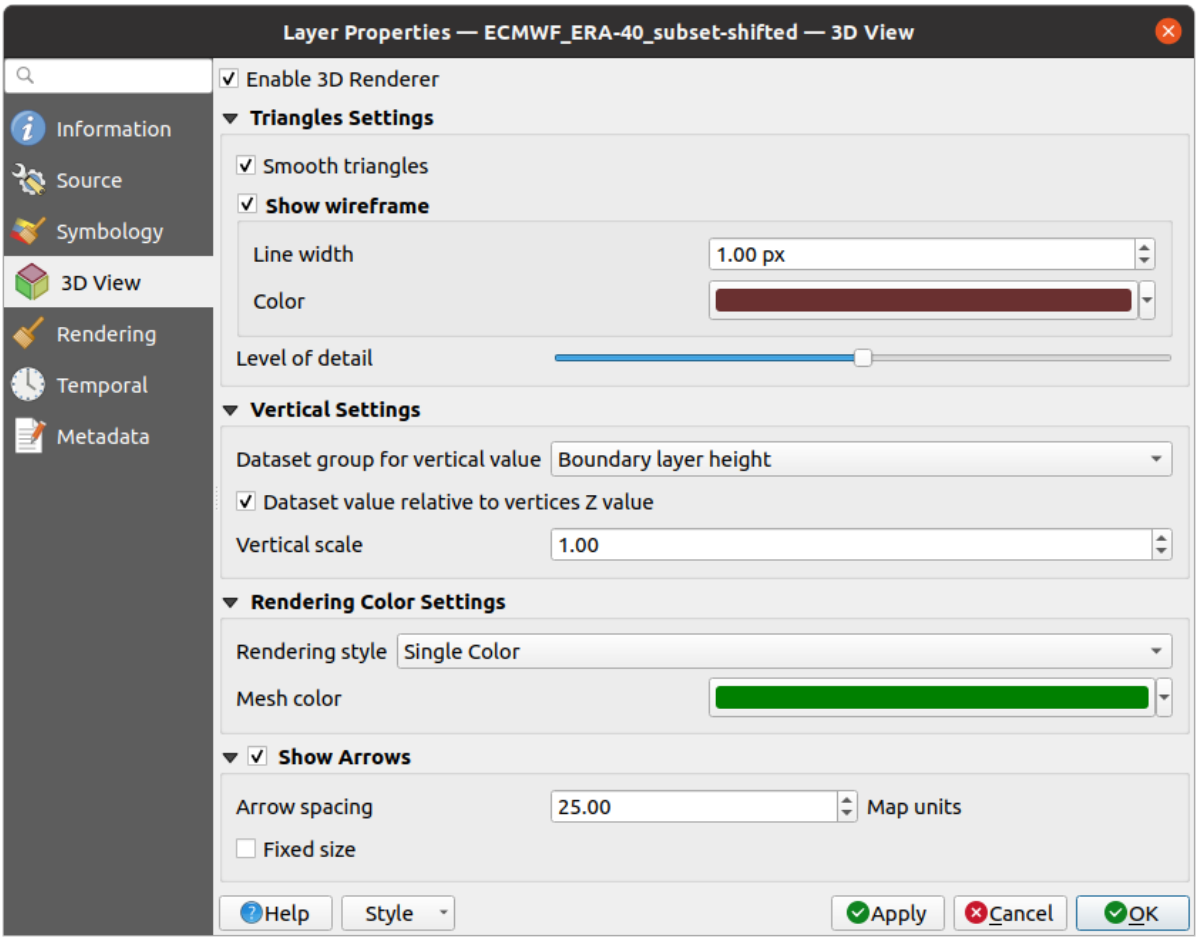


図 18.10: メッシュデータセットの 3D プロパティ

3D **レンダラを有効にする** にチェックを入れると、以下のプロパティを編集できます：

- **三角形設定** では、
  - **スムース三角形** ：3D レンダリングを改善するため、隣接する三角形の間の角をスムージングする
  - **ワイヤフレームを表示** ： **線幅** と **色** を設定可能
  - **詳細水準**（*Level of detail*） ：レンダリングするメッシュレイヤをどれだけ **簡素化** するかを制御します。スライダーの右端は元のメッシュで、左に行くほどメッシュレイヤは簡略化され、ディテールがより少なくなります。このオプションは、 **レンダリング** タブで **メッシュの簡素化** オプションを有効にした場合のみ利用可能です。

- **垂直設定 は、垂直成分のふるまいを制御します。**
  レンダリングされる三角形の頂点について、
  - **垂直方向のデータセットグループ** ：メッシュの垂直成分に使用するデータセットグループ
  - **Z 値に対する相対値** ：データセットの値を絶対 Z 座標とみなすか、または頂点のネイティブな Z 値からの相対座標とみなすか
  - **鉛直スケール** ：データセットの Z 値に適用するスケーリング係数
- **レンダリング色設定** ： **レンダリングスタイル** は、**等高線シンボロジ** で設定したカラーランプシェーダ（*2D* **等高線カラーランプシェーダ**）に基づく色、または、 **メッシュの色** に関連付けられた **単一色**
- **矢印を表示** ： **ベクトルの *2D* レンダリング** で使用されるデータセットグループと同じデータセットに基づいて、メッシュレイヤデータセットの 3D エンティティ上に矢印を表示します。矢印は 2D の色設定を使って表示されます。また、 **矢印の間隔** の定義や、 **固定サイズ** とするか大きさでスケーリングするかを定義できます。矢印は重なり合うことができないため、この矢印の間隔設定は矢印の最大サイズも定義します。

18.3.5 レンダリングプロパティ

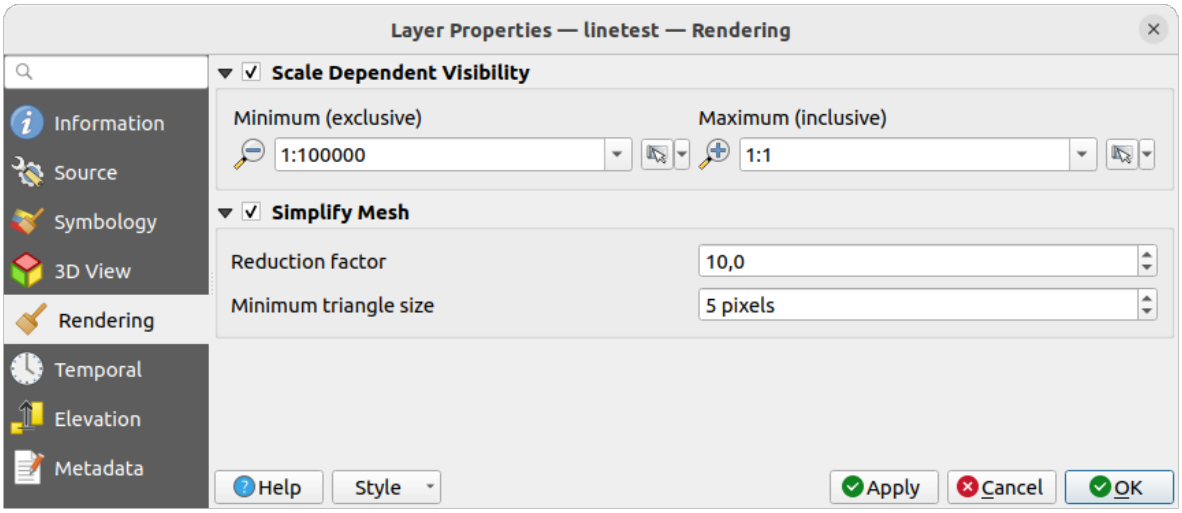


図 18.11: メッシュレンダリングのプロパティ

縮尺に応じた表示設定 グループボックスでは、 **最大縮尺** と **最小縮尺** を設定し、メッシュ要素を見せる縮尺の範囲を定義することができます。この範囲の外では隠されます。 現在のキャンバススケールを使う ボタンを使用すると、現在のマップキャンバススケールを可視範囲の境界として使用することができます。詳しくは **表示縮尺セレクタ** を参照してください。

注釈: レイヤの縮尺に応じた表示は、**レイヤ** パネルからも有効にすることもできます: レイヤを右クリックし、コンテキストメニューから **レイヤの縮尺表示を設定** を選択します。

メッシュレイヤは数百万の面を持つことがあるため、レンダリングには非常に時間がかかることがあります。これは特に、面が小さすぎて表示はできないものの、全ての面がビュー内にある場合に顕著です。レンダリングを高速化するためにメッシュレイヤを簡素化すると、さまざまな **詳細水準**（*level of detail*）を表す 1 つまたは複数のメッシュが生成され、QGIS にどの詳細水準のメッシュレイヤをレンダリングさせるかを選択できます。簡素化されたメッシュは三角形の面のみとなることに留意してください。

レンダリング タブで **メッシュの簡素化** にチェックを入れると、以下が設定できます：

- **削減係数** ：簡素化されたメッシュの連続するレベルの生成を制御します。例えば、ベースメッシュに 5 百万個の面があり、削減係数が 10 の場合、最初の簡素化されたメッシュはおよそ 500,000 個の面、2 番目は 50,000 個の面、3 番目は 5000 個の面、といった具合になります。削減係数が大きいほどより単純なメッシュ（つまり、より大きなサイズの三角形）に早く到達し、詳細水準の数も少なくなります。
- **最小三角形サイズ** ：表示が許可される三角形の平均サイズ（ピクセル単位）です。メッシュの平均サイズがこの値よりも小さい場合には、より低い詳細水準のメッシュのレンダリングに切り替わります。

18.3.6 時系列プロパティ

時系列 タブは、時間経過に伴うレイヤのレンダリングを制御するオプションを提供します。これは、有効化されたデータセットグループの時系列値を動的に表示できるようになります。このような動的なレンダリングには、マップキャンバスで **時系列ナビ** を有効にする必要があります。

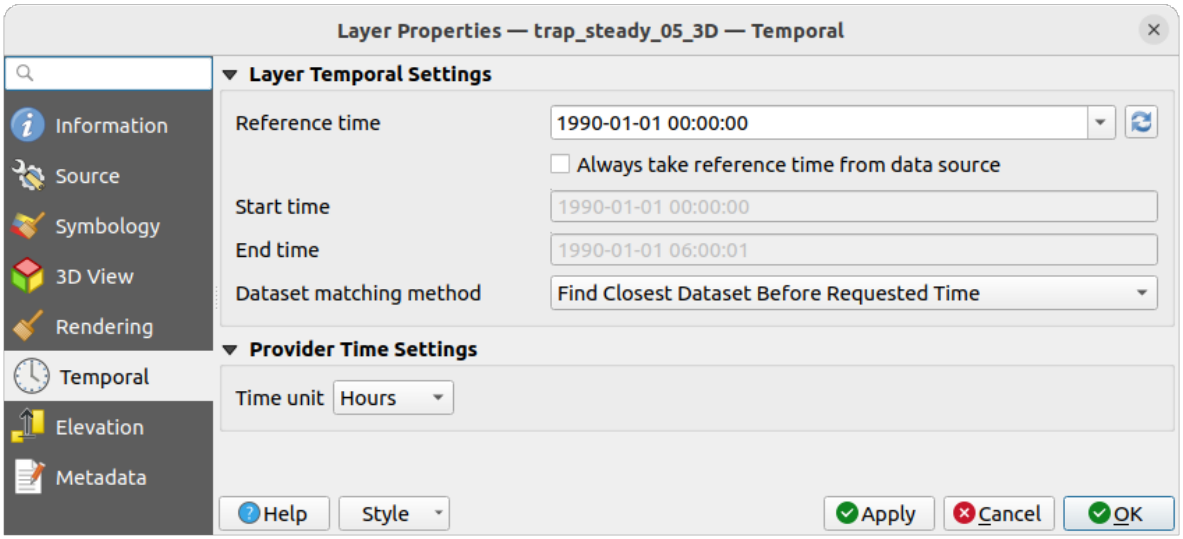


図 18.12: メッシュの時系列プロパティ

レイヤ時間設定

- **参照時間** は、データセットグループの絶対的な日時としての基準です。デフォルトでは、QGIS はソースレイヤを解析し、レイヤのデータセットグループ内の最初の有効な参照時間を返します。これが利用できない場合は、この値はプロジェクトの時間範囲によって設定されるか、現在の日付にフォールバックされます。考慮すべき **開始時刻** と **終了時刻** は、データセット内部のタイムスタンプのステップに基づいて計算されます。

カスタム **参照時間**（そして時間範囲）を設定すること、プロバイダから再読み込み ボタンを使って変更を元に戻すことができます。**常にデータソースから参照時間を取得** をチェックすると、レイヤーが再読み込みされたりプロジェクトが再開されたりするたびに、時間プロパティがファイルから更新されます。

- **データセット検索方法**：指定された時間に表示するデータセットを決定します。選択肢は、**時間内に最も近いデータセットを見つける** と **時間内に最も近いデータセットを見つける（前後）** の 2 つです。

プロバイダ時間設定

- **時間単位** は、元データから抽出されるか、ユーザーが定義します。この値は、マップの時系列ナビゲーションの際に、メッシュレイヤの速度をプロジェクト内の他のレイヤと揃えるために使用されます。サポートしている単位は、 **秒** 、 **分** 、 **時** 、および **日** です。

18.3.7 標高プロパティ

標高 タブには、 *3D* **マップビュー** 内のレイヤの標高プロパティと、 *profile tool charts* 内のレイヤの標高をコントロールするオプションがあります。具体的には次が設定できます:

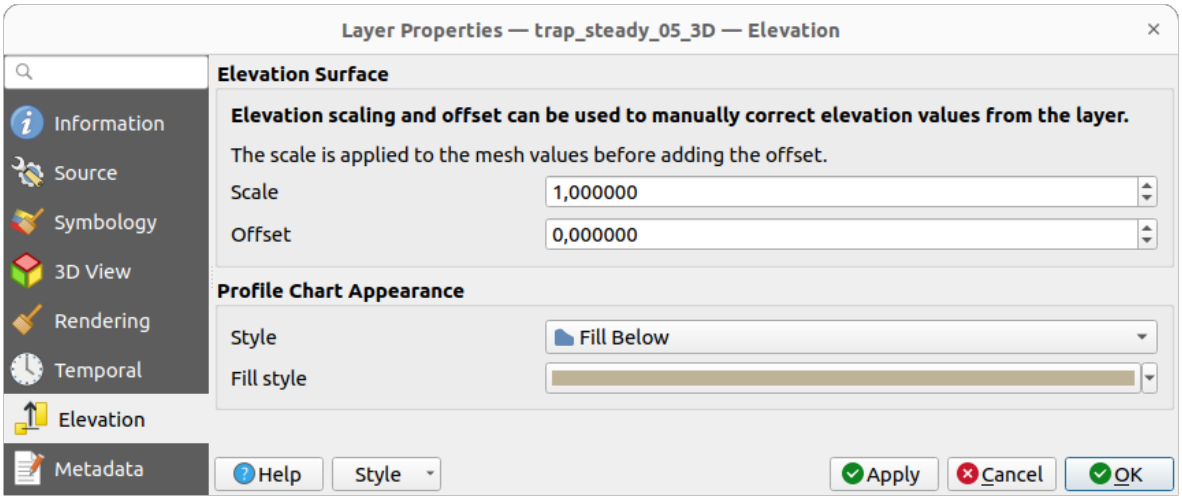


図 18.13: メッシュ標高プロパティ

- **標高サーフェス**: メッシュレイヤの頂点の Z 値を地形の標高としてどのように解釈するかを指定します。**スケール** 係数と **オフセット** を適用することができます。
- **断面図の外観**: 断面図を描く際に、メッシュ標高が使用するレンダリングの **スタイル** を制御します。次のように設定できます：
  - a profile *Line* with a *line style* applied
  - **下を塗りつぶす** と対応する **塗りつぶしスタイル** を持つ表面

18.3.8 メタデータプロパティ

 **メタデータ** タブには、レイヤに関するメタデータレポートの作成・編集オプションがあります。詳細は **メタデータ** を参照してください。

18.4 メッシュレイヤを編集する

QGIS では、ゼロからまたは既存のメッシュレイヤを元に **メッシュレイヤを作成する** ことができます。新しいレイヤのジオメトリを作成/変更することができ、後からデータセットを割り当てることができます。既存のメッシュレイヤを編集することもできます。編集にはフレームのみのレイヤが必要なため、初めに関連するデータセットを削除するか（必要であれば利用できることを確認してください）、（ジオメトリのみの）レイヤのコピーを作る必要があります。

注釈: QGIS ではメッシュレイヤの辺をデジタイズすることはできません。作成できるメッシュ要素は頂点と面のみです。また、サポートされているすべてのメッシュフォーマットが QGIS で編集できるわけではありません（permissions を参照してください）。

18.4.1 メッシュデジタイジングツールの概要

ベースメッシュレイヤ要素を操作・編集するには、以下のツールが利用できます。

表 18.4: メッシュデジタイジングのツール

Label	目的	場所
 現在の編集	全レイヤまたは選択レイヤを同時に保存・ロールバック・編集キャンセルするツールへのアクセス	**デジタイジング** ツールバー
 編集モード切り替え	レイヤの編集モードをオン／オフする	**デジタイジング** ツールバー
 レイヤ編集内容を保存	レイヤに行なった変更を保存する	**デジタイジング** ツールバー
 元に戻す	最後の変更を元に戻す - `Ctrl+z`	**デジタイジング** ツールバー
 やり直す	最後に取り消した操作をやり直す - `Ctrl+Shift+Z`	**デジタイジング** ツールバー
 高度なデジタイズツール(	**高度なデジタイズパネル** をオン／オフする	**高度なデジタイズ** ツールバー
 面と点を再インデックス(	メッシュ要素のインデックスを再作成し、番号を付け直して最適化する	**メッシュ** メニュー
 メッシュ要素をデジタイ	頂点と面を選択／作成する	**メッシュデジタイジング** ツールバー
 ポリゴンによるメッシュ要	描いたポリゴンに重なる頂点と面を選択する	**メッシュデジタイジング** ツールバー
 式によるメッシュ要素選択	式を使って頂点と面を選択する	**メッシュデジタイジング** ツールバー
 Transform Vertices Coordir	選択された頂点の座標を変更する	**メッシュデジタイジング** ツールバー
 選択されたジオメトリで分	線形ジオメトリを使用して面を分割し、Z 値を制約する	**メッシュデジタイジング** ツールバー

18.4.2 Z 値割り当てロジックを探検する

メッシュレイヤを編集モードにすると、マップキャンバスの右上に **頂点の Z 値** ウィジェットが開きます。デフォルトでは、その値は **設定** ▸ **オプション** ▸ **デジタイズ** タブで設定された **デフォルトの Z 値** に対応します。選択された頂点がある場合、ウィジェットは選択された頂点の平均 Z 値を表示します。

編集中は、新しい頂点に **頂点の Z 値** が割り当てられます。カスタム値を設定することもできます。ウィジェットを編集して `Enter` を押すと、デフォルト値が上書きされ、デジタイジング処理でこの新しい値が使用されます。ウィジェットの  アイコンをクリックすると、その値がオプションのデフォルト値にリセットされます。

割り当て規則

新しい頂点を **作成** するとき、その Z 値の定義はメッシュレイヤのアクティブな選択とその位置によって異なる場合があります。次の表は様々な組み合わせを表しています。

表 18.5: 新しい頂点への Z 値の割り当ての表

頂点の作成	メッシュレイヤに選択された頂点はあるか？	割り当てた値の元	割り当てられた Z 値
他に接続していない「自由な」頂点 面または面の辺	いいえ	**頂点の Z 値**	デフォルトまたはユーザー定義
		高度なデジタイズパネル（z ウィジェットが Locked 状態の場合）	Locked 状態のときは z ウィジェット
	はい	**頂点の Z 値**	選択された頂点の平均
辺にある頂点	---	メッシュレイヤ	辺の頂点からの内挿
面上の頂点	---	メッシュレイヤ	面の頂点からの内挿
2D ベクタ地物にスナップした頂点	---	**頂点の Z 値**	デフォルトまたはユーザー定義
3D ベクタ頂点にスナップした頂点	---	ベクタレイヤ	頂点
3D ベクタセグメントにスナップした頂点	---	ベクタレイヤ	ベクタセグメントに沿った内挿

注釈: **高度なデジタイズパネル** が有効で、メッシュ要素が選択されていない場合、**頂点の Z 値** ウィジェットは非アクティブになります。前者の z ウィジェットが Z 値の割り当てを行います。

既存の頂点の Z 値を変更する

頂点の Z 値を変更するのに最も簡単な方法は:

1. 1 つまたは複数の頂点を選択します。**頂点の Z 値** ウィジェットは選択範囲の平均の高さを表示します。
2. ウィジェットの値を変更します。
3. `Enter` を押します。入力した値が頂点に割り当てられ、次の頂点のデフォルト値になります。

頂点の Z 値を変更するもう一つの方法は、Z 値の能力を持つベクタレイヤ地物の上に頂点を移動してスナップすることです。複数の頂点が選択されている場合は、この方法で Z 値を変更することはできません。

Transform mesh vertices ダイアログでは、選択した頂点の Z 値を（X 座標または Y 座標とともに）変更することもできます。

18.4.3 メッシュ要素を選択する

メッシュ要素をデジタイズ を使う


メッシュ要素をデジタイズ ツールをアクティブにします。要素にカーソルを合わせるとハイライトされ、選択できるようになります。

- 頂点の上でクリックするとそれが選択されます。
- 面や辺の中心にある小さな四角をクリックすると、それが選択されます。つながっている頂点も選択されます。逆に、辺や面のすべての頂点を選択すると、その要素も選択されます。
- 重なっている要素を選択するには矩形をドラッグします（選択された面にはすべての頂点が含まれます）。完全に含まれる要素だけを選択したい場合は `Alt` キーを押します。
- 選択範囲に要素を追加するには、要素を選択した状態で `Shift` を押します。
- 要素を選択範囲から外すには `Ctrl` を押して再度選択します。選択が解除された面は、その頂点の選択も解除されます。

ポリゴンによるメッシュ要素選択 を使う

:meshSelectPolygon| ポリゴンによるメッシュ要素選択 ツールをアクティブにする:

- メッシュジオメトリの上にポリゴンを描きます（左クリックで頂点を追加、`Backspace` で最後の頂点を元に戻す、`Esc` でポリゴンを中止し、右クリックで有効にする）。部分的に重なっている頂点や面が選択されます。完全に含まれる要素のみを選択したい場合は、描画中に `Alt` キーを押してください。
- ベクタレイヤの地物のジオメトリ上で右クリックし、ポップアップするリストから選択すると、メッシュレイヤの頂点や面が部分的に重なっていても選択されます。完全に含まれる要素のみを選択するには、描画中に `Alt` を使用します。
- 選択範囲に要素を追加するには、要素を選択した状態で `Shift` を押します。
- 選択範囲から要素を削除するには、選択ポリゴンの上に描画しながら `Ctrl` を押します。

式によるメッシュ要素選択

メッシュ要素を選択するもう一つのツールは 
式によるメッシュ要素選択 です。これを押すと、ツールの **式によるメッシュ要素選択ダイアログ** が開き、次のことができます:

1. 選択する方法を選ぶ:
   - **頂点による選択**: 入力された式を頂点に適用し、合致したものとそれに関連する辺/面を返します
   - **面による選択**: 入力された式を面に適用し、合致したものとそれに関連する辺/頂点を返します
2. 選択式を記述する。*Meshes* **グループ** で利用可能な関数は、選択された方法に応じてフィルタリングされます。

3. クエリを実行するには、以下の選択範囲の動作を設定し、押します:

   - **選択**: レイヤ内の既存の選択範囲を置き換える
   - **現在の選択に追加する**
   - **現在の選択範囲から除去する**

18.4.4 メッシュ要素を変更する

頂点を追加する

メッシュレイヤに頂点を追加するには:

1. メッシュ要素をデジタイズ ボタンを押す
2. マップキャンバスの右上に **頂点の Z 値** ウィジェットが表示されます。この値を後続の頂点に割り当てたい Z 座標に設定します
3. ダブルクリックします:

   - 面の外：「フリー頂点」、つまりどの面にもリンクされていない頂点を追加します。この頂点はレイヤが編集モードの時、赤い点で表示されます。
   - 既存の面の辺: 辺に頂点を追加し、触れた面を新しい頂点に接続された三角形に分割します。
   - 面の内側：新しい頂点と周囲の頂点と結んだ三角形に面を分割します。

面を追加する

メッシュレイヤに面を追加するには:

1. メッシュ要素をデジタイズ ボタンを押す
2. マップキャンバスの右上に **頂点の Z 値** ウィジェットが現れます。この値を後続の頂点に割り当てたい Z 座標に設定する。
3. 頂点にカーソルを合わせ、その横に表示される小さな三角形をクリックする。
4. カーソルを次の頂点の位置に移動する; 既存の頂点にスナップするか、左クリックして新しい頂点を追加します。
5. 上記と同じように、面に対して好きなだけ頂点を追加する。Backspace ボタンを押すと最後の頂点を元に戻すことができる。
6. マウスを動かしている間、面の形状を示すラバーバンドが表示されます。ラバーバンドが緑で表示されている場合、その面は有効であり、右クリックでメッシュに追加できます。赤で表示されている場合、その面は有効ではなく（例えば、自己交差している、既存の面や頂点と重なっている、穴を作っているなど）、追加できません。いくつかの頂点を元に戻し、ジオメトリを修正する必要があります。
7. :kbd:`Esc' を押すと面のデジタイジングを中止します。

8. 右クリックして面を検証する。

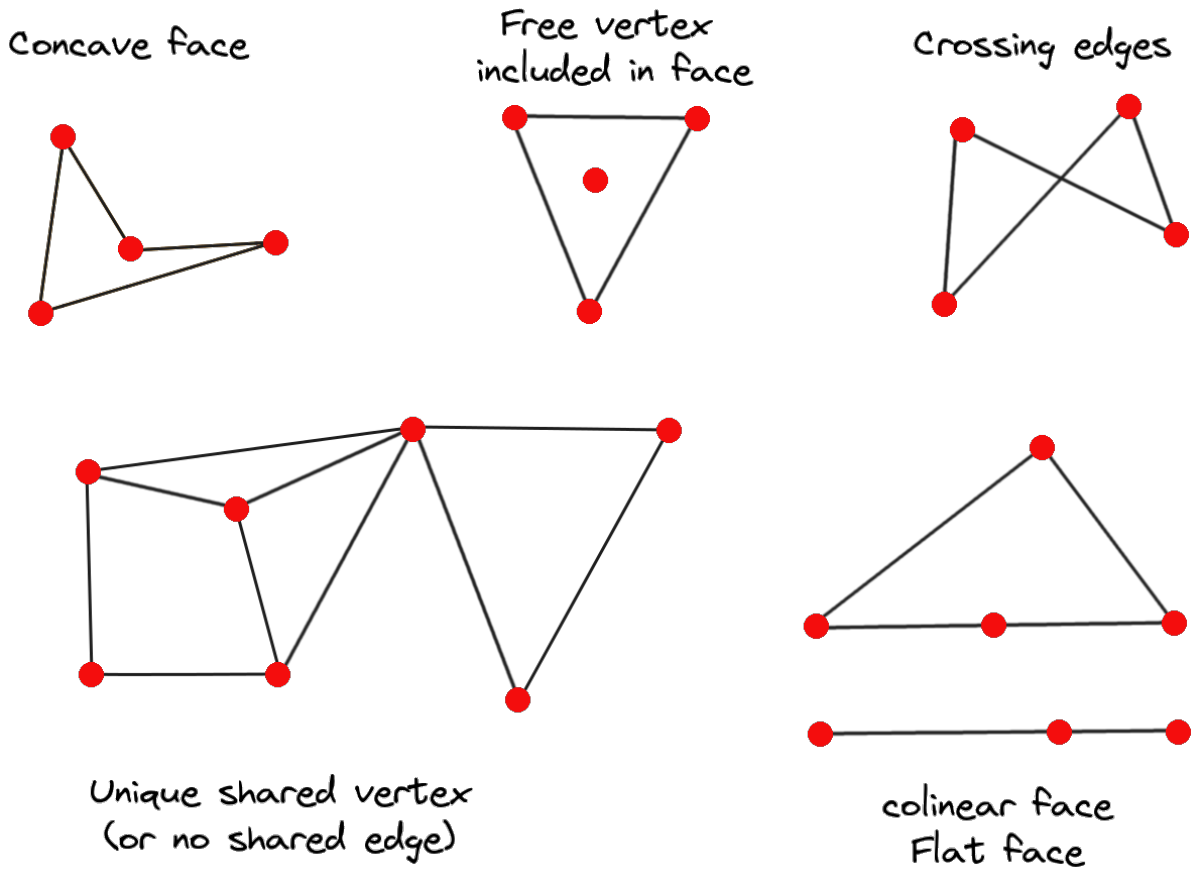


図 18.14: 無効なメッシュの例

メッシュ要素を削除する

1. **対象の要素を選択する**
2. メッシュ要素をデジタイズ ツールを有効にする
3. 右クリックして選択する:
   - **選択した頂点を削除し、穴を埋める** または Ctrl+Del を押す: 頂点及びリンクされた面を削除し、隣接する頂点から三角形を作って穴を埋めます
   - **選択した頂点を削除し、穴を埋めない** または Ctrl+Shift+Del を押す：頂点及びリンクされた面を削除し、穴を埋めない
   - **選択した面を削除する** 又は Shift+Del を押す: 面を削除しますが頂点は残します

これらのオプションには、アイテムを選択しなくてもカーソルを合わせてコンテクストメニューからアクセスできます。

メッシュ要素を移動する

メッシュレイヤの　頂点と面を移動するには:

1. **対象の要素を選択する**
2. メッシュ要素をデジタイズ ツールを有効にする
3. 要素の移動を開始するには、頂点または面/辺の重心をクリックします
4. カーソルを目的の位置に移動します（ベクタ地物へのスナップがサポートされています）。
5. 新しい位置が **無効メッシュ** を生成しない場合、移動した要素は緑色で表示されます。この位置でもう一度クリックすると解除されます。頂点がすべて選択されている面は平行移動され、それに応じて隣接する面の形も変更されます。

メッシュの頂点を変形する

頂点座標の変換 ツールは、式を使って X、Y 及び／又は Z 座標を編集することで、頂点を移動させるより高度な方法を提供します。

1. 座標を編集したい頂点を選択します
2. 頂点座標の変換 を押します。ダイアログが開き、選択された頂点の数が表示されます。選択範囲から頂点を追加したり削除したりすることができます。
3. 変更したいプロパティに応じて、*X* **座標**、*Y* **座標**、*Z* **値** のいずれかをチェックする必要があります。
4. 次に、数値または式（ 式ダイアログ を使用）でターゲット位置をボックスに入力します
5. 選択された頂点の座標をインポート を押すと、1 つの頂点が選択されるたびに、X、Y、Z のボックスが自動的にその座標で満たされます。頂点を個別に調整する便利で素早い方法です。
6. *Preview Transform* を押すと、頂点の新しい位置がシミュレートされ、変形したメッシュがプレビューされます。
   - プレビューが緑色であれば、変形されたメッシュは有効であり、変形を適用することができます。
   - プレビューが赤色の場合、変形されたメッシュは無効で、修正されるまで変形を適用できません。
7. **変形を適用** を押して、選択した頂点の座標を変更します。

メッシュジオメトリを整形する

コンテキストメニュー

1. メッシュ要素をデジタイズ を有効にします
2. メッシュアイテムを選ぶ、若しくは
3. メッシュ要素にカーソルを合わせ、ハイライトさせます。

4. 右クリックすると、次が可能になります:

   - **アイテムを削除する**
   - **選択された面を分割** (**現在の面を分割**): カーソルを合わせている面、または選択されている四角形のメッシュ面を 2 つの三角形に分割します
   - *Delaunay Triangulation with Selected vertices*: 選択された自由頂点を用いて三角形の面を作ります。
   - *Refine Selected Face(s)* (*Refine Current Face*): 各辺の中央に追加された頂点に基づいて、面を 4 つの面に分割します（三角形は三角形に、四角形は四角形になります）。また、新しい頂点に接続された隣接する面を三角形分割します。

辺マーカー


メッシュ要素をデジタイズ がアクティブなときに辺の上にカーソルを置くと、辺がハイライトされ、その辺を操作することができます。コンテキストによって、以下のマーカーが使用できます:

- 辺の中点にある **正方形**: クリックすると、両端の頂点が選択されます。
- 両側の二つの面がマージできるときの **十字**: クリックするとエッジが削除され、それらの面がマージされます。
- 辺が 2 つの三角形の間にある場合は、**円**: クリックすると辺が反転します。つまり、その辺を面の他の 2 つの「自由な」頂点に接続します。

選択されたジオメトリで強制 ツール


選択されたジオメトリで強制 ツールは、ラインジオメトリを使用してブレークラインを適用する高度な方法を提供します。ブレークラインはメッシュにラインに沿った辺を強制的に持たせます。操作が完了すると、ブレークラインは永続的なものとは見なされないことに注意してください; 結果として生じる辺はもう制約として機能せず、他の辺のように変更することができます。これは、例えば、川の堤防や道路の盛土の境界のように、正確な線でメッシュレイヤーを局所的に修正するために使うことができます。

1. 
選択されたジオメトリで強制 ツールを有効にします
2. 次のいずれかで「強制する線」として使用するジオメトリを示します:

   - マップキャンバスのライン又はポリゴン地物から選択: ベクタ地物上で右クリックし、コンテキストメニューのリストから選択します。
   - メッシュ枠上に引かれた仮想の線: 左クリックで頂点を追加し、右クリックで検証します。頂点の Z 値は **頂点の Z 値** ウィジェット、または **高度なデジタイズパネル** がオンの場合は z ウィジェットで設定します。線がメッシュの頂点や 3D ベクトル地物の頂点やセグメントにスナップされた場合、新しい頂点はスナップされた要素の Z 値をとります。


選択されたジオメトリで強制 ツールのドロップダウンメニューから設定できるオプションに依存する方法で、ラインジオメトリまたはポリゴンの境界に重なるメッシュ面が影響を受けます:

- **辺の交点に座標点を追加**: このオプションを指定すると、強制する線が辺と交差するたびに新しい頂点が追加されます。このオプションは、各面が出会うたびに線に沿って分割することにつながります。

  このオプションがないと、遭遇した面は削除され、既存の頂点と強制する線の頂点だけを持つ三角形分割から得られる面で置き換えられます（強制する線と交差する境界辺にも新しい頂点が追加されます）。

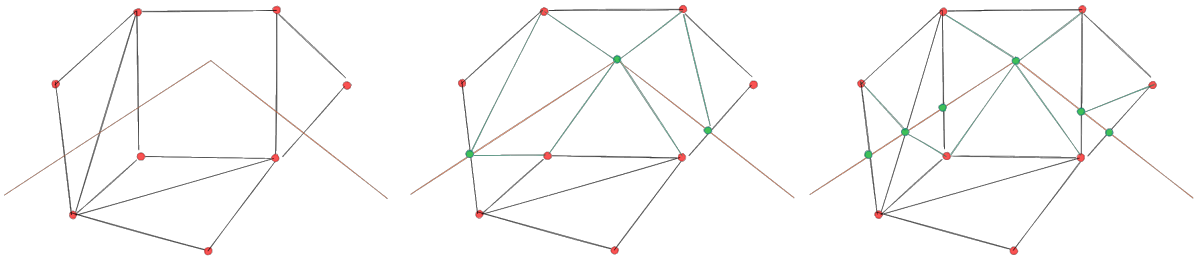


図 18.15: ラインジオメトリを使用したフォースメッシュ - エッジの交点に新しい頂点がない場合（中央）とある場合（右）の結果

- *Z* **値を内挿**: 新しい頂点の Z 値をどのように計算するかを設定します。これは:
  - **メッシュ** 自身: 新しい頂点の Z 値は、その頂点が含まれる面の頂点から内挿されます
  - または **強制する線**: 線が 3D ベクタ地物または描画された線で定義されている場合、新しい頂点の Z 値はそのジオメトリから導かれます。2D ライン地物の場合、新しい頂点の Z 値は **頂点の** *Z* **値** となります。
- **許容範囲**: 既存のメッシュ頂点が許容範囲よりも線に近い場合、線上に新しい頂点を作成せず、代わりに既存の頂点を使用します。この値は **縮尺済みメートル** または **地図単位** で設定することができます（詳細は **単位セレクタ** を参照）。

18.4.5 メッシュの再インデックス化

QGIS は編集中、また元に戻す/やり直す操作を素早く行うために、削除された要素のために空の場所を保持します。これは、メモリ使用量の増加や非効率的なメッシュ構造を引き起こす可能性があります。**|menuselection:`メッシュ -->` |meshReindex| 面と点を再インデックス化** ツールは、このような穴を取り除き、面と頂点のインデックスをリナンバし、連続的である程度合理的な順序になるように設計されています。これにより、面と頂点の関係が最適化され、計算効率が向上します。

注釈: **面と点を再インデックス化** ツールはレイヤを保存し、元に戻す/やり直すのスタックをクリアし、ロールバックを無効にします。

18.5 メッシュ計算機

メッシュ メニューの先頭にある **メッシュ計算機** ツールを使用すると、既存のデータセットグループに対して算術演算や論理演算を行い、新しいデータセットグループを生成できます（ 図 18.16 参照）。

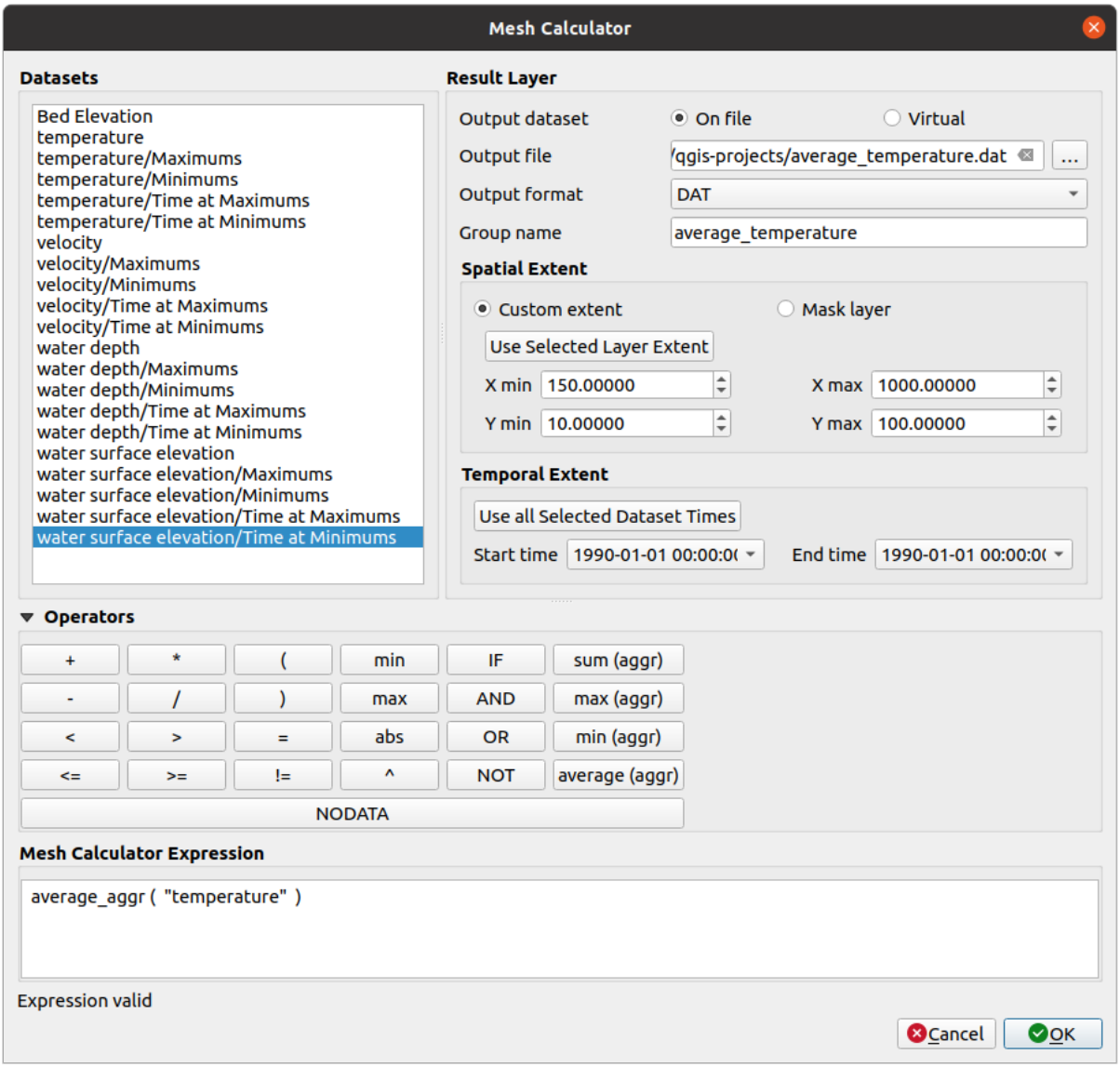


図 18.16: メッシュ計算機

データセット のリストには、アクティブなメッシュレイヤ内の全てのデータセットグループが含まれます。データセットグループを式で使用するには、リスト内でグループ名をダブルクリックします。すると、**メッシュ計算機の式** フィールドにグループが追加されます。続いて演算子ボタンを使用したりボックスに演算子を直接入力して、計算式を構築してください。

ラスタレイヤ では、出力レイヤのプロパティを設定できます：

- **ディスクに書き込まないオンザフライ・ラスタを作成**
  - これにチェックを入れない場合、出力結果は通常の新しいファイルとしてディスク上に保存されます。 **出力ファイル** のパスと、 **出力形式** を指定する必要があります。

  - チェックを入れた場合、メッシュレイヤに新しいデータセットグループが追加されます。データセットグループの値はメモリには保存されておらず、各データセットはメッシュ計算機に入力された計算式で必要となった時に計算されます。この仮想のデータセットはプロジェクトに保存され、レイヤの **ソース** プロパティタブから必要に応じて削除したり、ファイルに永続化させたりすることができます。

  どちらの場合でも、出力データセットグループの **グループ名** を指定する必要があります。

- **空間範囲** は計算で考慮する範囲で、以下の設定があります：
  - **カスタム範囲** ：座標の *X* **最小値** 、 *X* **最大値** 、 *Y* **最小値** 、 *Y* **最大値** を手動で入力するか、既存のデータセットグループから抽出します（範囲に使用したいデータセットをリストで選択し、**選択レイヤの領域を使用** ボタンを押すと上記の座標フィールドが入力されます）
  - プロジェクトのポリゴンレイヤによる定義（ **マスクレイヤ** ）：ポリゴン地物のジオメトリを使用してメッシュレイヤのデータセットを切り抜きます
- **時系列範囲** ：データセットが考慮する時間範囲を **開始時刻** と **終了時刻** のオプションで設定します。これは、データセットグループの既存の時間ステップの中から選択します。また、 **全ての選択データセット時間を使用** ボタンを押して、時間の全範囲を設定することもできます。

演算子 セクションには利用可能なすべての演算子があります。演算子をメッシュ計算機の式ボックスに追加するには、それに応じたボタンをクリックします。算術計算（ `+` 、 `-` 、 `*` など）や統計関数（ `min` 、 `max` 、 `sum (aggr)` 、 `average (aggr)` など）が利用できます。条件式（ `=` 、 `!=` 、 `<` 、 `>=` 、 `IF` 、 `AND` 、 `NOT` など）は、偽の場合は 0 、真の場合は 1 を返すので、他の演算子や関数と共に使うことができます。 `NODATA` 値も式中で使用できます。

メッシュ計算機の式 ウィジェットには実行する式が表示され、式を編集することができます。

第19章　ベクタタイルの操作

19.1 ベクタタイルとは？

ベクタタイルは地理データのパケットであり、Web 上で転送するためにあらかじめ定義されたほぼ正方形の「タイル」にパッケージ化されています。これらは、事前にレンダリングされたラスタ地図タイルとベクタ地図タイルを組み合わせたものです。ベクタタイルサーバは、あらかじめレンダリングされた地図画像の代わりに、各タイルの境界でクリップされたベクタ地図データを返します。クリップされたタイルはベクタタイルサービスのズームレベルを表すもので、ピラミッドアプローチに由来しています。この構造を使用することで、タイル化されていないベクタ地図に較べてデータ転送を削減できます。現在のマップビュー範囲内の、現在のズームレベルのデータのみを転送すれば済むからです。また、タイル化されたラスタ地図との比較では、ベクタデータは通常、レンダリングされたビットマップよりもはるかに小さいため、データ転送も大幅に削減されます。ベクタタイルにはスタイル設定の情報は割り当てられていないため、データを表示するためには QGIS で地図作成スタイルを適用する必要があります。

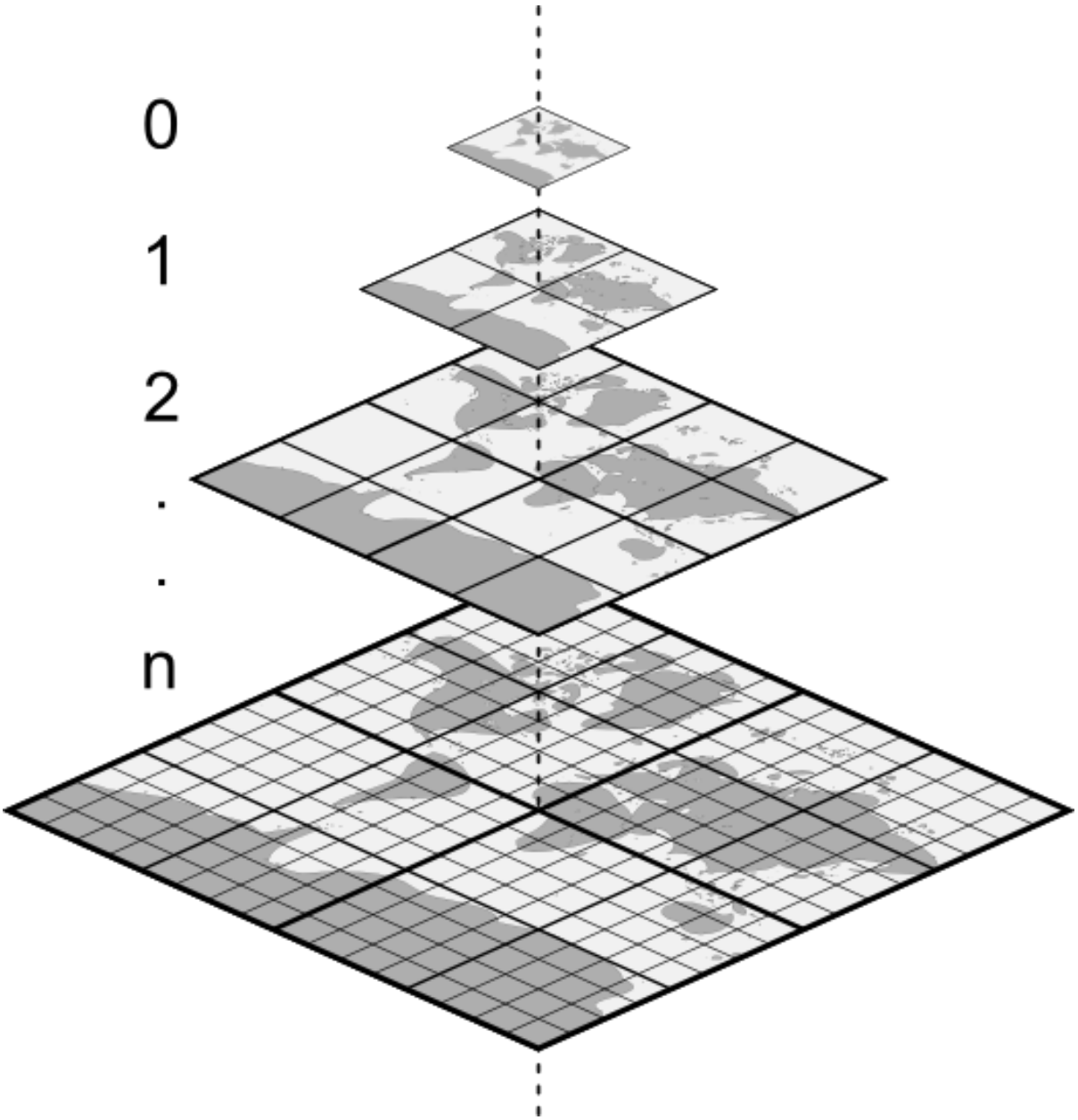


図 19.1: ズームレベルを持つベクタタイルのピラミッド構造

19.2 サポートする形式

ベクタタイルのサポートは以下のとおりです：

- リモートソース（HTTP/S） - XYZ テンプレートあり - 例： `type=xyz&url=http://example.com/{z}/{x}/{y}.pbf`
- ローカルファイル - XYZ テンプレートあり - 例： `type=xyz&url=file:///path/to/tiles/{z}/{x}/{y}.pbf`
- ローカルの MBTiles データベース - 例： `type=mbtiles&url=file:///path/to/file.mbtiles`

ベクタタイルデータセットを QGIS にロードするには、**データソースマネージャ** ダイアログの **ベクタタイル** タブを使用します。詳細は **ベクタタイルサービスを利用する** を参照してください。

19.3 ベクタタイルデータセットのプロパティ

19.3.1 情報プロパティ

情報 タブは読み取り専用で、現在のレイヤの要約された情報やメタデータをさっと掴むことができる興味深い場所です。提供される情報には、以下のものがあります：

- レイヤのプロバイダからの情報：名前、URI、ソースタイプとパス、ズームレベルの数
- 空間参照システム（CRS）：CRS の名前、単位、投影法、精度、参照（静的か動的か）
- **入力されたメタデータ** からの情報：アクセス、領域、リンク、連絡先、履歴など

19.3.2 シンボロジプロパティとラベルプロパティ

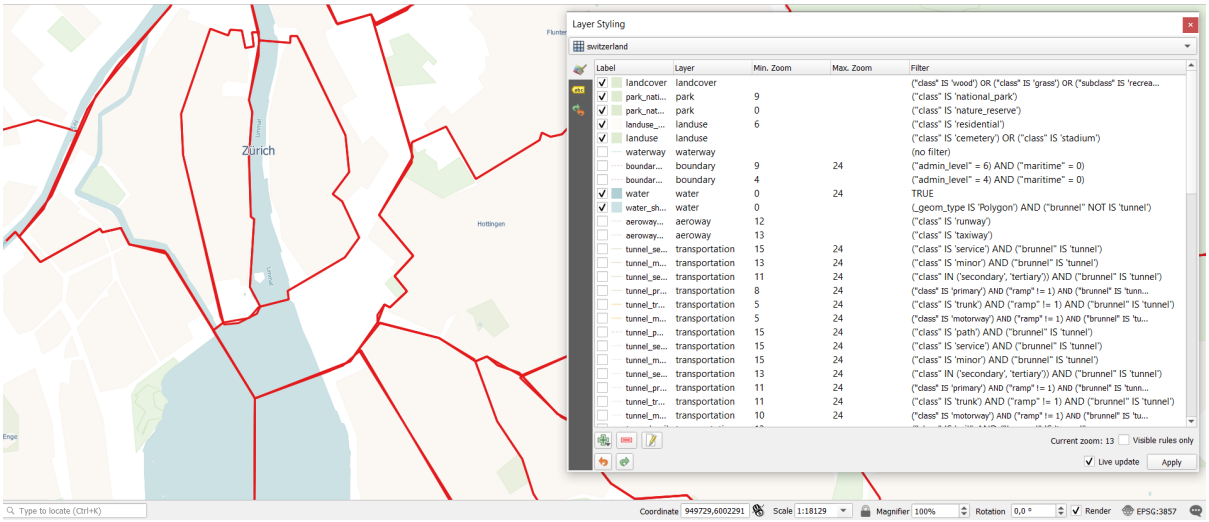


図 19.2: ベクタタイルレイヤのシンボロジ

ベクタタイルはポイント、ライン、ポリゴンのジオメトリで構成されているため、それぞれでシンボルを使用できます。地図作成スタイルを適用するには、 **ベクタタイル接続** を作成するときに *StyleURL* を使用する必要があります。シンボルは *OK* ボタンをクリックした後すぐに **シンボロジ** タブに表示されます。

独自の地図作成スタイルを作成するために、地物に **ルール** のセットを定義し、スタイルとラベルに適用することができます。 図 19.2 では、OpenStreetMap の `landuse` レイヤのスタイルとラベル付けを設定しています。ここでは、 suburb クラスに対して設定を行っています。マップを見やすくするため、ほとんどのルールの選択を解除しています。

下部には **現在のズーム** が表示されています。 **表示できるルールのみ** オプションにチェックを入れると、指定したズームレベルで表示されるルールのみにリストをフィルタリングできます。これにより、複雑な

ベクタスタイルの設定作業や、問題のあるルールの特定が容易になります。スタイルとラベル付けは、ズームレベルに依存するようにできます。

スタイルをインポートするオプションもあります。スタイルは以下の形式で与えます：

- *QML* ファイル（*QML-QGIS* **スタイルファイル形式** ）
- *MapBox GL JSON* スタイル設定ファイル

19.3.3 メタデータプロパティ

 **メタデータ** タブには、レイヤに関するメタデータレポートの作成・編集オプションがあります。詳細は **メタデータ** を参照してください。

第 20 章　点群の操作

20.1 点群入門

点群とは？

点群とは、多数のデータ点（最大数十億、数兆）からなる空間の 3 次元画像です。各点は x、y、z 座標を持ちます。キャプチャ方法にもよりますが、点群には通常、色値や強度など、キャプチャに由来した追加属性もあります。これらの属性は、例えば、異なる色で点群を表示するために使用することができます。QGIS では、点群を使って風景（または別の空間）の 3 次元画像を生成することができます。

サポートしている形式

QGIS は、Entwine Point Tile (EPT) と LAS/LAZ のデータ形式をサポートしています。点群データを扱う場合、QGIS は常に EPT でデータを保存します。EPT は、共通のフォルダーに保存される複数のファイルで構成された保存形式です。データに素早くアクセスできるように、EPT はインデックスを使います。EPT 形式の詳細については、entwine homepage を参照してください。

データが LAS または LAZ 形式の場合、QGIS は初回に読み込む時に EPT に変換します。ファイルの大きさによっては、これに時間がかかる場合があります。この処理では、LAS/LAZ ファイルがあるフォルダに、 `ept_ + name_LAS/LAZ_file` というスキームでサブフォルダーが作られます。そのようなサブフォルダーがすでに存在する場合、QGIS は EPT をすぐに読み込みます（読み込み時間の短縮につながります）。

知っておくべきこと

QGIS では（まだ）点群を編集することはできません。点群を操作したい場合は、オープンソースの点群処理ツールである `CloudCompare <https://www.cloudcompare.org/>`_ を使うことができます。また、Point Data Abstraction Library （PDAL - GDAL に似ている）は点群を編集するオプションを提供しています（PDAL はコマンドラインのみです）。

データ点の数が多いため、QGIS では点群の属性テーブルを表示することはできません。しかし、  *Identify tool* は点群をサポートしているので、1 つのデータポイントであってもすべての属性を表示することができます。

20.2 点群のプロパティ

点群レイヤの **レイヤプロパティ** ダイアログは、レイヤとそのレンダリングに関する一般的な設定を提供します。また、レイヤの情報も提供します。

レイヤプロパティ ダイアログにアクセスするには：

- **レイヤ** パネルでレイヤをダブルクリックするか、右クリックしてコンテキストメニューから **プロパティ**... を選択する；

- レイヤを選択した状態で、 **レイヤ**　**レイヤのプロパティ**... メニューを選ぶ

点群 **レイヤプロパティ** ダイアログには次のセクションがあります：

[1] **レイヤスタイルパネル** からも利用可能です

注釈: 点群レイヤのプロパティのほとんどは、プロパティダイアログの下部にある **スタイル** メニューを使って `.qml` ファイルに保存したり、そこから読み込んだりすることができます。詳細は **レイヤのプロパティの保存および共有** を参照してください。

20.2.1 情報プロパティ

情報 タブは読み取り専用で、現在のレイヤの要約された情報やメタデータをさっと掴むことができる興味深い場所です。提供される情報には、以下のものがあります：

- 名前、ソースパス、最終保存日時と大きさ、使用されたプロバイダなどの一般
- レイヤのプロバイダに基づいた：領域と点の数
- 空間参照系：名前、単位、方法、精度、参照（即ち、静的か動的のいずれか）
- プロバイダからのメタデータ：作成日、バージョン、データフォーマット、縮尺 X/Y/Z…
- **メタデータ** タブから選ばれた（編集可能な）：アクセス、領域、リンク、連絡先、履歴…

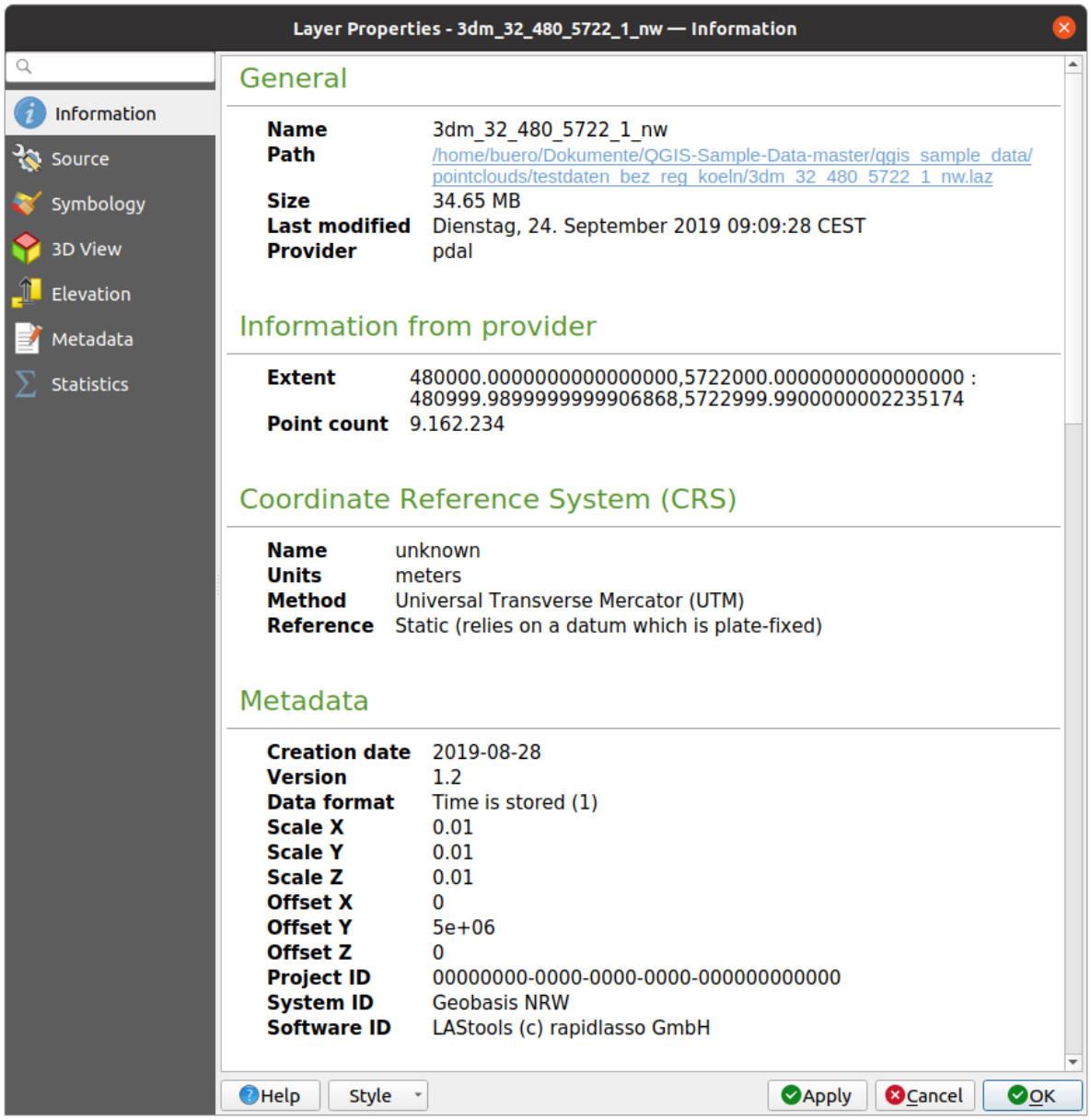


図 20.1: 点群情報タブ

20.2.2 ソースプロパティ

ソース タブでは、点群レイヤの基本情報を確認・編集することができます：

- **設定**: プロジェクト（レイヤパネル、式、印刷レイヤ凡例…）でレイヤを識別するのに使われるレイヤ名をレイヤファイル名と異なるものに設定します
- **設定された** *CRS*： ここではレイヤに割り当てられている **座標参照系** を変更することができます。ドロップダウンリストで最近使用したものを選択するか、set Projection Select CRS ボタンをクリックします (参照 **座標参照系セレクタ**)。この処理は、レイヤに適用されている CRS が誤っている場合や、何も適用されていない場合にのみ使用します。

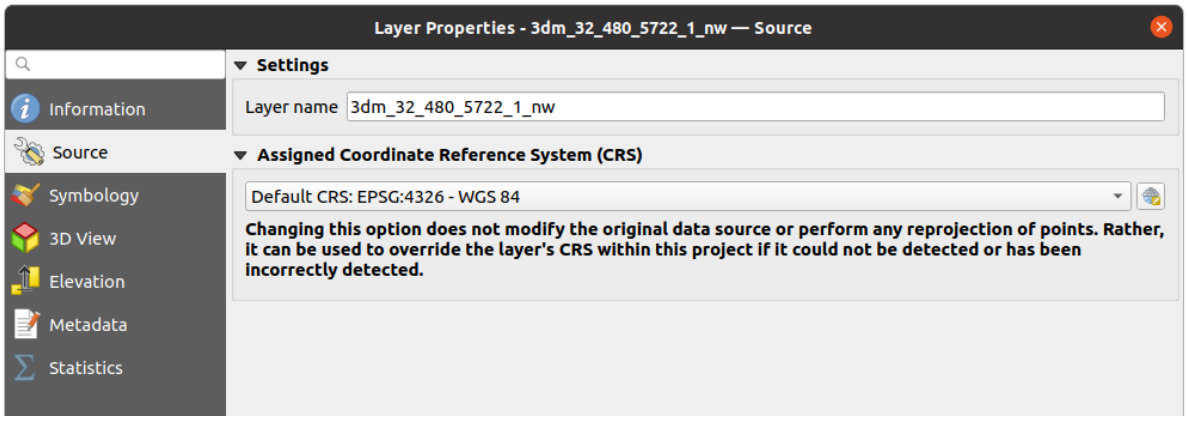


図 20.2: 点群のソースタブ

20.2.3 シンボロジプロパティ

シンボロジ タブでは、点群のレンダリングの設定を行います。上部には、異なる地物レンダラーの設定があります。下部には、レイヤ全体の一般的な設定と、地物レンダラーに適用される設定があります。

地物レンダリングタイプ

点群のレンダリングにはさまざまなオプションがあり、 **シンボロジ** タブの上部にあるドロップダウンメニューで選択することができます（ 図 20.3 を参照）：

- **範囲のみ**： データの範囲のバウンディングボックスのみを表示します；データの範囲を概観するのに便利です。普通通り、 **シンボル ウィジェット** を使用することで、ボックスのプロパティ（色、ストローク、不透明度、サブレイヤなど）を設定することができます。
- **ランプによる属性**: データはカラーグラデーション上に描画されます。**ランプによる属性** を参照
- *RGB*： 赤、緑、青の色値を用いてデータを描画します。*RGB* **レンダラー** を参照
- **分類**： データは分類ごとに異なる色で描画されます。**分類レンダラー** を参照してください

点群が読み込まれると、QGIS はロジックに従って最適なレンダラーを選択します：

- データセットに色情報（赤、緑、青属性）が含まれている場合、RGB レンダラーが使われます
- そうではなく、データセットに `Classification` 属性が含まれていれば、分類レンダラーが使われます
- そうでなければ、Z 属性に基づくレンダリングにフォールバックします

点群の属性がわからない場合は、 **統計量の出力 タブ** を参照すると、点群にどの属性が含まれていて、どの範囲に値があるのかの概要を知ることができます。

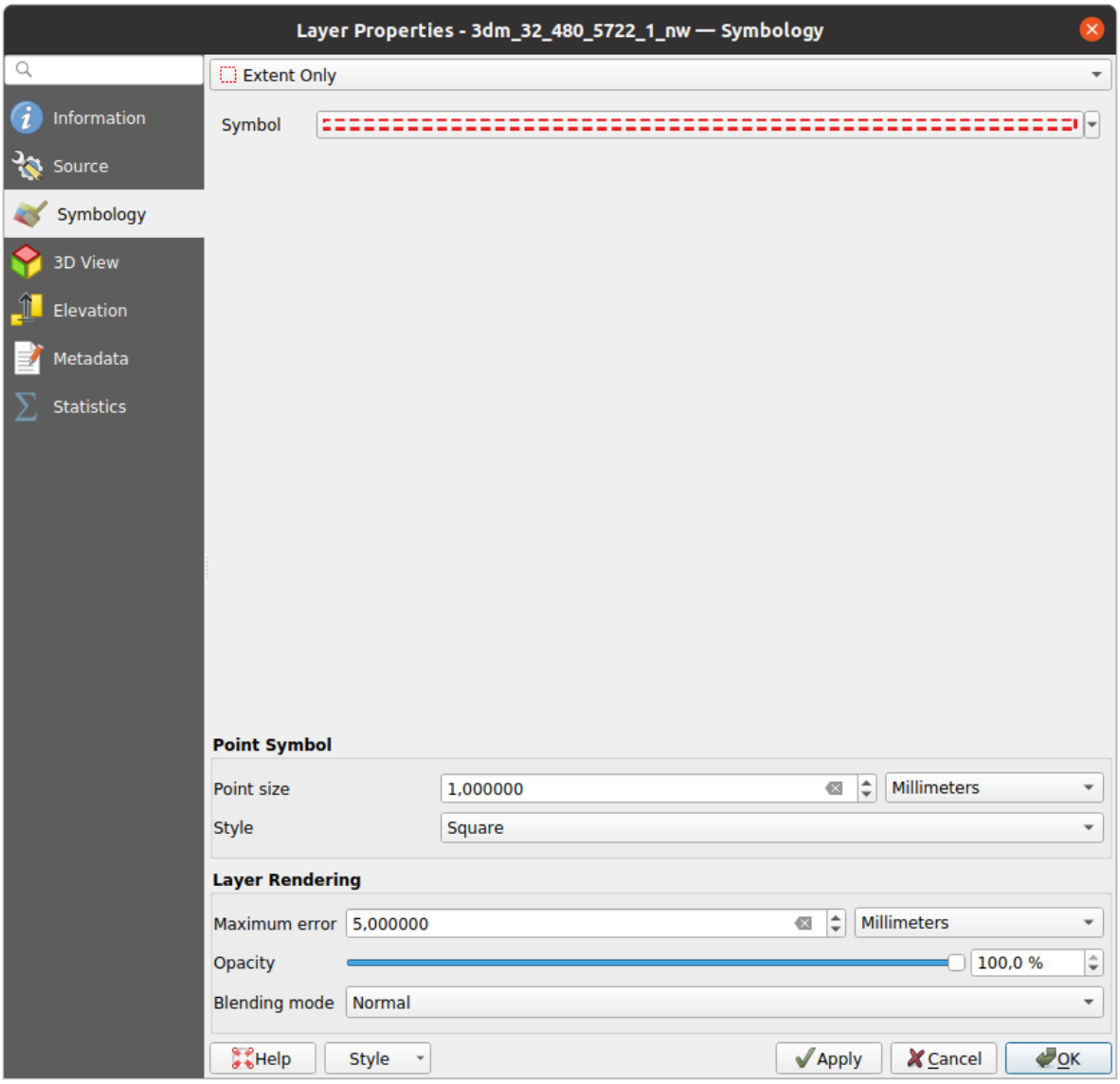


図 20.3: 点群シンボロジタブ

ランプによる属性

ランプによる属性 では、データをカラーグラデーション上の数値で表示することができます。このような数値は、例えば、既存の強度属性や Z 値にすることができます。最小値と最大値により、他の値は内挿によってカラーグラデーションに広がります。明確な値と特定の色への割り当ては「カラーマップ」と呼ばれ、表に示されます。様々な設定オプションがあり、それらは図の下に説明されています。

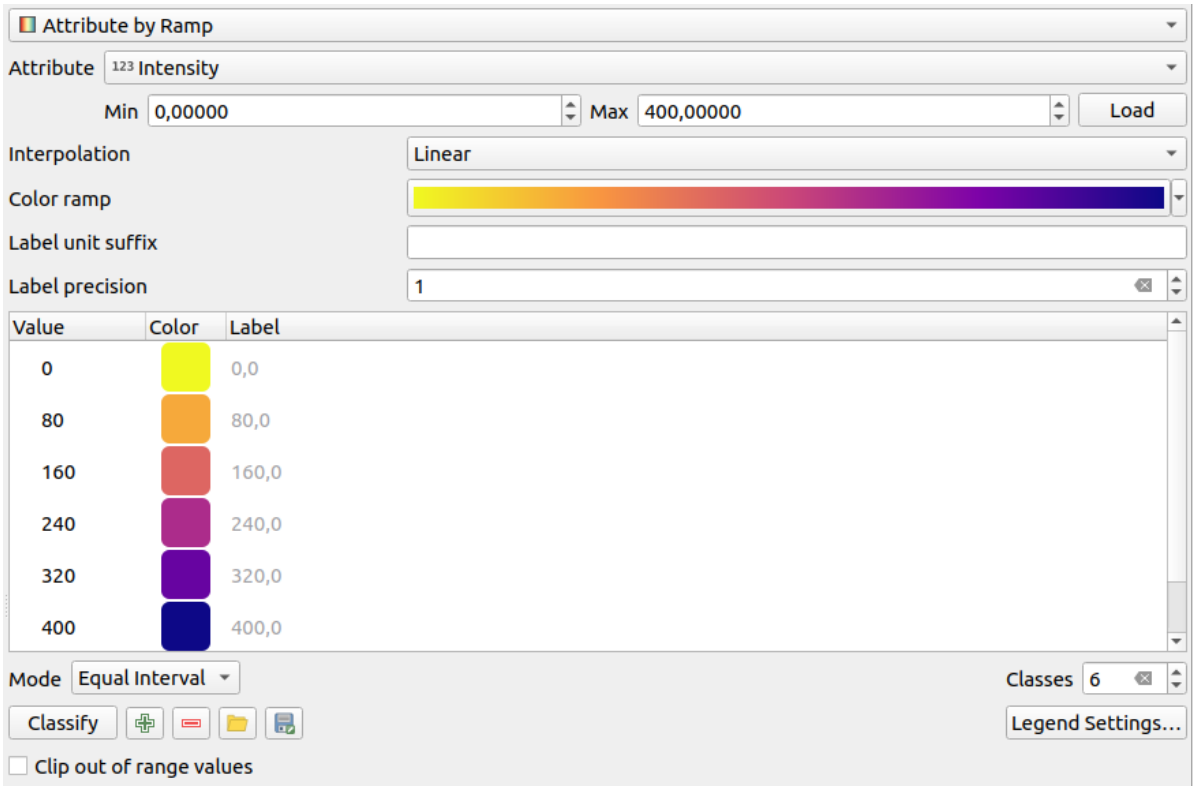


図 20.4: 点群シンボロジタブ：ランプによる属性

- **最小値** と **最大値** はカラーランプに適用される範囲を定義します： **最小値** の値はカラーランプの左端、 **最大値** の値は右端を表し、その間の値は内挿されます。デフォルトでは、QGIS は選択された属性から最小値と最大値を検出しますが、変更することもできます。値を変更したら、**読み込み** ボタンをクリックしてデフォルトに戻すことができます。
- **内挿** の項目は、値にどのように色が割り当てられるかを定義します：
  - **離散値** （<= シンボルが **値** カラムのヘッダに表示されます）：ラスタの色には、その値以上で最も近いカラーマップのエントリを使用します
  - **線形** ピクセル値の上下にあるカラーマップの項目から線形補間された色であり、データセット値毎にユニークな色に対応します
  - **正確な**（= シンボルが **値** カラムのヘッダに表示されます）：カラーマップエントリの値に等しい値を持つピクセルだけに色が適用されます。その他の値のピクセルはレンダリングされません。
- **カラーランプ** ウィジェットは、データセットに割り当てるカラーランプを選択するのに役立ちます。**このウィジェット** と同様に、新しいウィジェットを作ったり、現在選択されているウィジェットを編集して保存することができます。
- **ラベルの単位の接尾辞** は、凡例内で値の後ろにラベルを追加します。また、 **ラベルの精度** は表示する小数点以下の桁数を制御します。

分類の **モード** は、クラス間でどのように値が分布するかを定義できます：

- **連続的**： クラス番号と色はカラーランプのストップから取得されます; 限界値はカラーランプのストップの分布に従って設定されます (ストップに関する詳細は **カラーランプの設定** を参照してください)。

- **等間隔分類**： クラスの数は行末の **クラス** フィールドによって設定されます; 限界値はクラスがすべて同じ大きさになるように定義されます。

クラスは自動的に決定され、カラーマップ表に表示されます。しかし、手動でこれらのクラスを編集することもできます：

- テーブルの **値** をダブルクリックするとクラス値を変更することができます
- guilabel:**色** 列をダブルクリックすると 色を選択 ウィジェットが開き、その値に適用する色を選択することができます
- **ラベル** 列をダブルクリックするとクラスのラベルを変更することができます
- カラーテーブルで選択された行の上で右クリックすると、 **色を変更**... と **不透明度を変更**... のコンテキストメニューが表示されます

テーブルの下には、**分類** でデフォルトのクラスに戻すか、手動で値を  追加 するか、表から選択した行を  削除 するオプションがあります。

カスタマイズされたカラーマップは非常に複雑になる可能性があるため、既存のカラーマップを  読み込む オプションや、他のレイヤで使用するために（ `txt` ファイルとして）  保存 するオプションもあります。

内挿 で **線形** を選択した場合は、以下の設定も可能です：

-  **範囲外の値を無視** デフォルトで、線形法はデータセットの値のうち、設定された **最小値** より小さな値（設定された **最大値** より大きな値）に対して最初のクラス（最後のクラス）の色を割り当てます。これらの値をレンダリングしたくない場合は、この設定をチェックしてください。
- **凡例の設定** は **レイヤ** パネルと **レイアウト凡例** に表示されます。カスタマイズはラスタレイヤと同じように動作します（詳細は **ラスタの凡例のカスタマイズ** を参照してください）。

RGB レンダラー

multibandColor| *RGB* レンダラーでは、点群から選択された 3 つの属性が赤、緑、青の成分として使用されます。属性に適切な名前が付けられている場合、QGIS は自動的に属性を選択し、各バンドの **最小値** と **最大値** を取得し、それに応じてカラーリングをスケーリングします。しかし、手動で値を変更することも可能です。

コントラスト 方法には、 **強調しない**、 **最小最大範囲に引き伸ばす**、 **最小最大範囲に引き伸ばしカット**、**最小から最大までの範囲以外は無視** の値が適用できます

注釈: **コントラスト** ツールはまだ開発中です。問題がある場合は、デフォルトの **最小最大範囲に引き伸ばす** を使用してください。

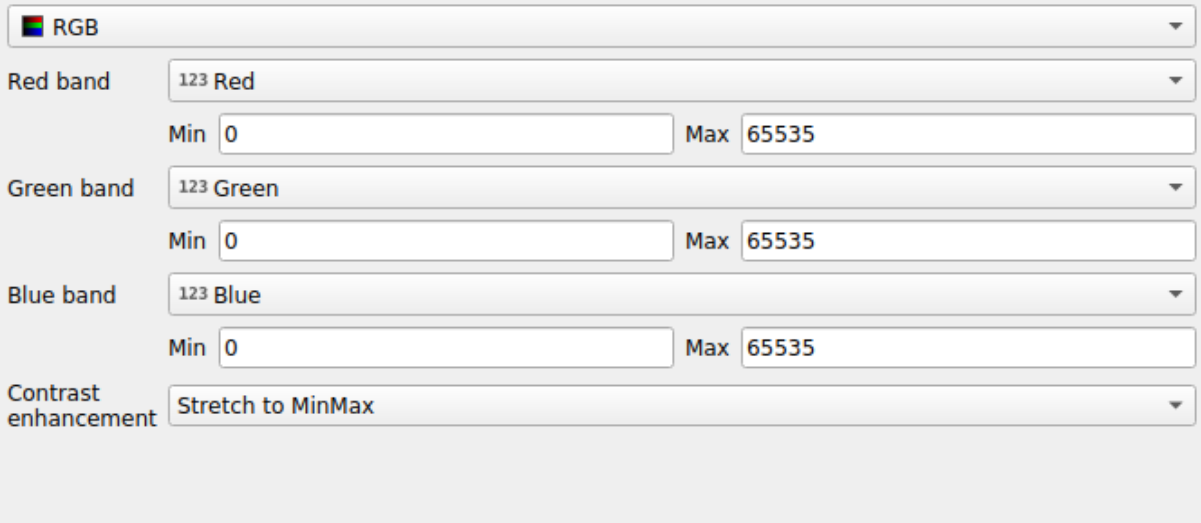


図 20.5: 点群 RGB レンダラー

分類レンダラー

分類 レンダリングでは、点群は属性に基づいて色分けされて表示されます。属性はどのようなタイプでも使用できます（数値、文字列、...）。点群データにはしばしば `Classification` というフィールドが含まれます。このフィールドには通常、後処理によって自動的に決定された植生などのデータが含まれます。**属性** を使うと、属性テーブルから分類に使用するフィールドを選択することができます。デフォルトでは、QGIS は LAS 仕様の定義を使用します（ASPRS ホームページ にある PDF の 'ASPRS 標準点クラス' の表を参照してください）。しかし、データはこのスキーマから逸脱している可能性があります。疑問がある場合は、データを受け取った人や機関に定義を問い合わせる必要があります。

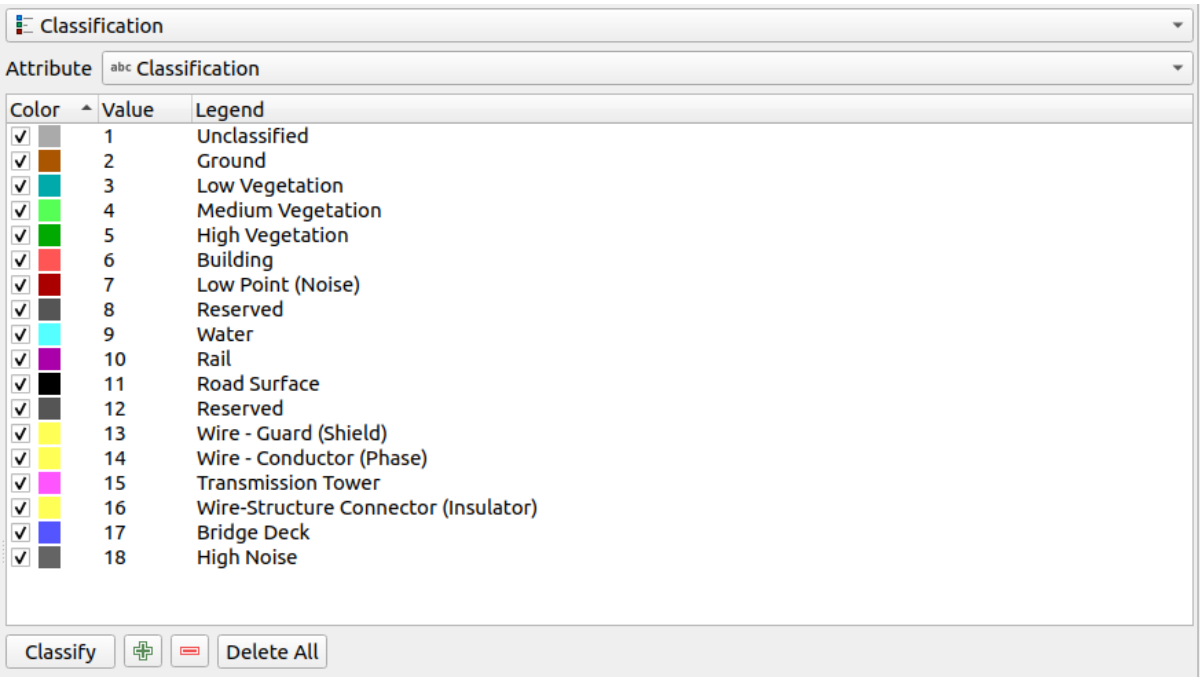


図 20.6: 点群分類レンダラー

表では、使用されたすべての値が対応する色と凡例とともに表示されます。各行の先頭には チェックボックスがあり、チェックを外すとその値はマップ上に表示されなくなります。テーブルをダブルクリックすると **色**、**値**、**凡例** を変更することができます（色については 色選択 ウィジェットが開きます)。

表の下にはボタンがあり、QGIS が生成するデフォルトクラスを変更することができます：

- guilabel:**分類** ボタンを使用すると、データを自動的に分類することができます：属性に出現し、テーブルにまだ存在しないすべての値が追加されます
- 追加 と 削除 を使うと、値を手動で追加または削除できます
- **すべて削除** は表からすべての値を取り除きます

ヒント: レイヤ パネルでは、レイヤのクラスリーフエントリの上で右クリックすると、対応する地物の可視性を素早く設定することができます。

点シンボル

点シンボル では、各データポイントを表示する大きさと単位（ミリメートル、ピクセル、インチなど）を設定することができます。点のスタイルとして **円** または *Square* が選択できます。

レイヤレンダリング

レイヤレンダリング セクションには、レイヤのレンダリングを変更するための以下のオプションがあります：

- **描画順**: 2 次元マップキャンバス上の点群のレンダリング順を Z 値に依存するかどうかを制御できます。次でレンダリングすることができます:
  - 点がレイヤに格納されている **デフォルト** 順、
  - :guilabel:`下から上`（Z 値が大きい点は低い点を覆い、真のオルソ写真のように見える)、
  - または、情景を下から見ているように見える **上から下**。
- **最大エラー**: 点群には通常、表示に必要な以上の点が含まれています。このオプションで、点群の表示をどの程度密にするか、または疎にするかを設定します（これは「点間の最大許容ギャップ」とも理解できます)。大きな数値（例：5 mm）を設定すると、点と点の間に目に見える隙間ができます。低い値（例：0.1mm）を設定すると、不必要な量の点をレンダリングすることになり、レンダリングが遅くなります（異なる単位を選択することができます)。
- **不透明度**： このツールでマップキャンバスの下にあるレイヤを表示させることができます。スライダーを使ってレイヤの可視性をニーズに合わせて調整します。また、スライダーの横のメニューで可視性のパーセンテージを正確に定義することもできます。
- **混合モード**： このツールで特殊なレンダリング効果を得ることができます。オーバーレイレイヤとその下のレイヤのピクセルは、**混合モード** で説明されている設定によって混合されます。

- **アイドーム照明**: これはマップキャンバスにシェーディング効果を適用し、より良い深度レンダリングを行います。レンダリングの品質は *draw order* プロパティに依存します。**デフォルト** の描画順では最適な結果が得られないかもしれません。以下のパラメータを制御することができます:
  - **強度**: コントラストを増加し、奥行き感を与えます
  - **距離**: 中心ピクセルからの使用ピクセルの距離を表し、エッジを太くする効果があります。

20.2.4 3D ビュープロパティ

 *3D* **ビュー** タブでは、3D マップでの点群のレンダリングの設定を行うことができます。

3D レンダリングモード

タブ上部のドロップダウンメニューから以下のオプションが選択できます：

- **レンダリングしない**: データは表示されません
- *2D* **シンボルに従う**: 3D の地物レンダリングを *2D* **に割り当てたシンボロジ** に合わせます
-  **単一色**： すべての点は属性に関係なく同じ **色** で表示されます
-  **ランプによる属性**： 指定された属性をカラーランプで補間し、それにマッチする色を地物に割り当てます。**ランプによる属性** を参照。
-  *RGB*： 地物に割り当てる赤、緑、青の色成分を設定するために、地物の異なる属性を使います。*RGB* **レンダラー** を参照。
-  **分類**: 属性に基づいてポイントを色で区別します。**分類レンダラー** を参照。

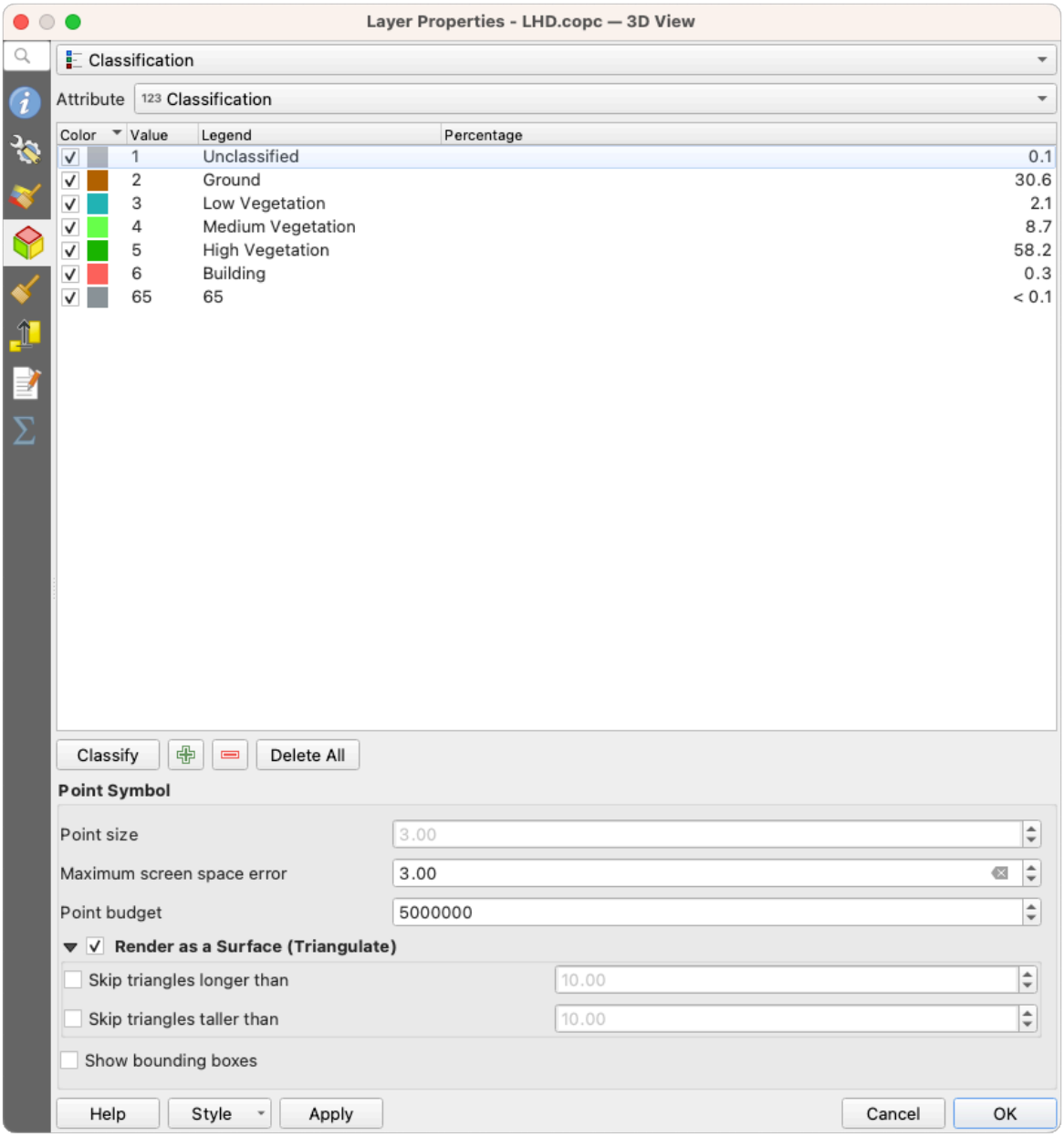


図 20.7: 点群 3D ビュータブと分類レンダラー

3D 点シンボル

3D **ビュー** タブの下部には **点シンボル** セクションがある。ここでは全てのレンダラに同じ、レイヤ全体の一般的な設定を行うことができる。以下のオプションがある：

- **点の大きさ**: 各データポイントを表示する大きさ（ピクセル単位で）を設定できます
- **最大画面誤差**： このオプションで、点群の表示を（ピクセル単位で）どの程度密にするか、または疎にするかを設定します。大きな数値（例 10）を設定すると、点と点の間に目に見える隙間ができます。低い値（例 0）を設定すると、不必要な量の点をレンダリングすることになり、レンダリングが遅くなります（詳細は **シンボロジ** **最大エラー** を参照）。

- *Point budget*： 長時間のレンダリングを避けるために、レンダリングされるポイントの最大数を設定できます
- **サーフェス（三角網）としてレンダリング** をチェックすると、3D ビューの点群レイヤを三角形分割によって得られた平面でレンダリングします。計算された三角形の寸法を制御できます：
  - **スキップする三角形の幅の下限**: 水平面で、三角形の一辺の長さの最大値を設定します
  - **スキップする三角形の高さの下限**: 垂直面で、三角形の一辺の高さの最大値を設定します
- **バウンディングボックスを表示**： 特にデバッグに便利で、階層内のノードのバウンディングボックスを表示します

20.2.5 レンダリングプロパティ

縮尺に応じた表示設定 グループボックスの下で、 **最大縮尺（含む）`と** *:guilabel:* `**最小縮尺（含まない）** を設定し、地物を可視とする縮尺の範囲を定義することができます。この範囲外では隠される。 現在のキャンバスが表示している縮尺を設定 ボタンを使うと、現在のマップキャンバスの縮尺を可視範囲の境界として使うことができます。詳しくは **表示縮尺セレクタ** を参照してください。

注釈: レイヤの縮尺に応じた表示は、**レイヤ** パネルからも有効にすることもできます: レイヤを右クリックし、コンテキストメニューから **レイヤの縮尺表示を設定** を選択します。

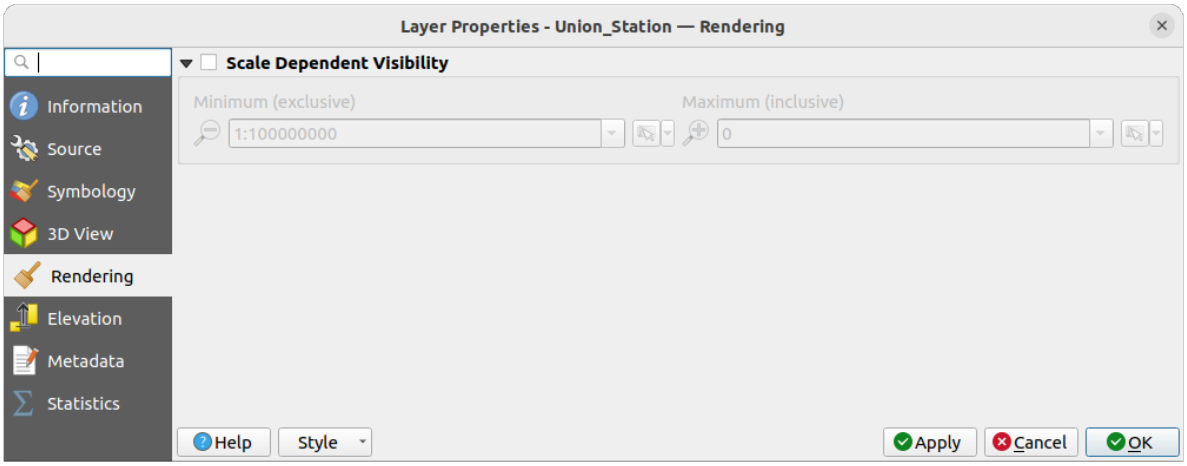


図 20.8: 点群レンダリングタブ

20.2.6 高さプロパティ

高さ タブでは、データの Z 値の補正を設定することができます。これは 3D マップでのデータの標高や、 *profile tool charts* での見栄えを調整するために必要かもしれません。以下のオプションがあります：

- **高さ** グループの下では:
  - **縮尺** を設定することができます: ここに `10` を入力すると、Z = 5 の点は `50` の高さで表示されます。
  - z レベルに対する **オフセット** を入力することができます。これは異なるデータソースの高さを一致させるのに便利です。デフォルトでは、データに含まれる最も低い z 値がこの値として使用されます。この値は行末の Refresh ボタンで元に戻すこともできます。
- **断面図の精度** の下にある **最大エラー** は、標高断面図でレンダリングされる点をどの程度密にするか、または疎にするかを制御するのに役立ちます。値を大きくすると、含まれる点が少なくなり、生成が速くなります。
- **断面図の外観** では、点の表示を制御できます：
  - **点の大きさ**: サポートされている単位（ミリメートル、地図単位、ピクセル、...）で、点をレンダリングするサイズ
  - **スタイル**: **円** または **四角** のどちらで点をレンダリングするか
  - 単一の **色** を断面図ビューに表示されている全ての点に適用する
  - **レイヤーの色に従う** をチェックすると、代わりに *2D* **シンボロジ** によって割り当てられた色で点を表示します
  - **距離に応じた不透明度を適用** は、プロファイル曲線から遠い点の不透明度を下げます

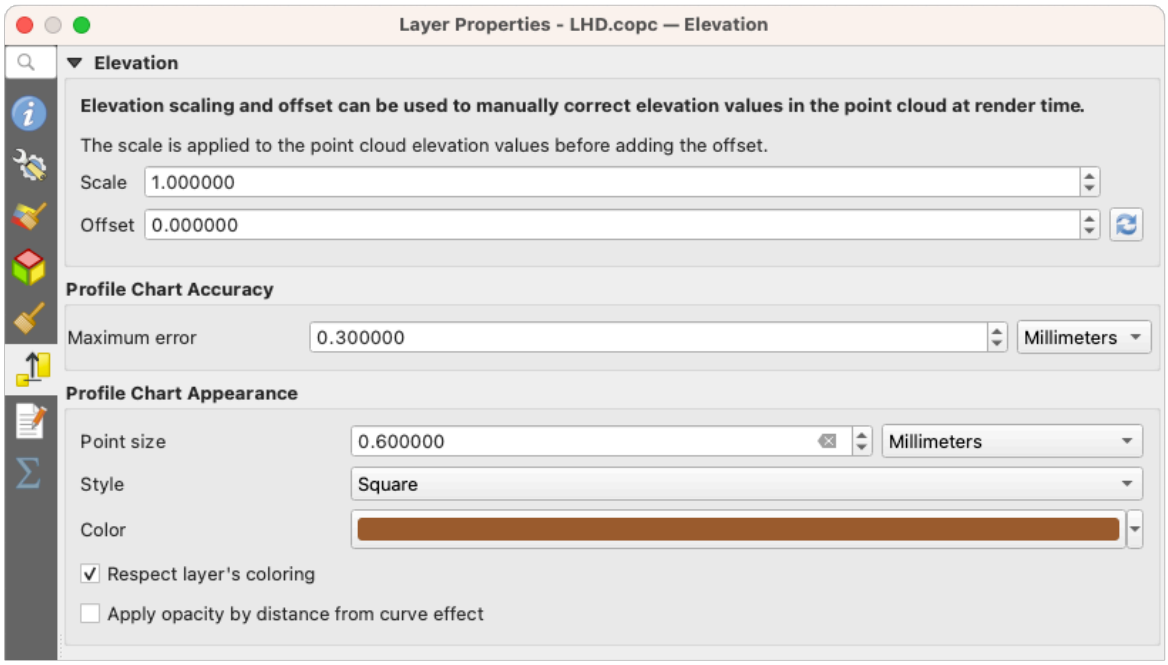


図 20.9: 点群高さタブ

20.2.7 メタデータプロパティ

メタデータ タブには、レイヤに関するメタデータレポートの作成・編集オプションがあります。詳細は **メタデータ** を参照してください。

20.2.8 統計量の出力プロパティ

統計量の出力 タブでは、点群の属性とその分布の概要を知ることができます。

一番上には **属性の統計量** というセクションがあります。ここには点群に含まれる全ての属性と、その統計値の一部が一覧表示されます： **最小値**、**最大**、**平均値**、**標準偏差**

もし *Classification* という属性があれば、下のセクションに別の表があります。ここには、その属性に含まれる全ての値と、その絶対的な **カウント** と相対的な % 存在度が一覧にされています。

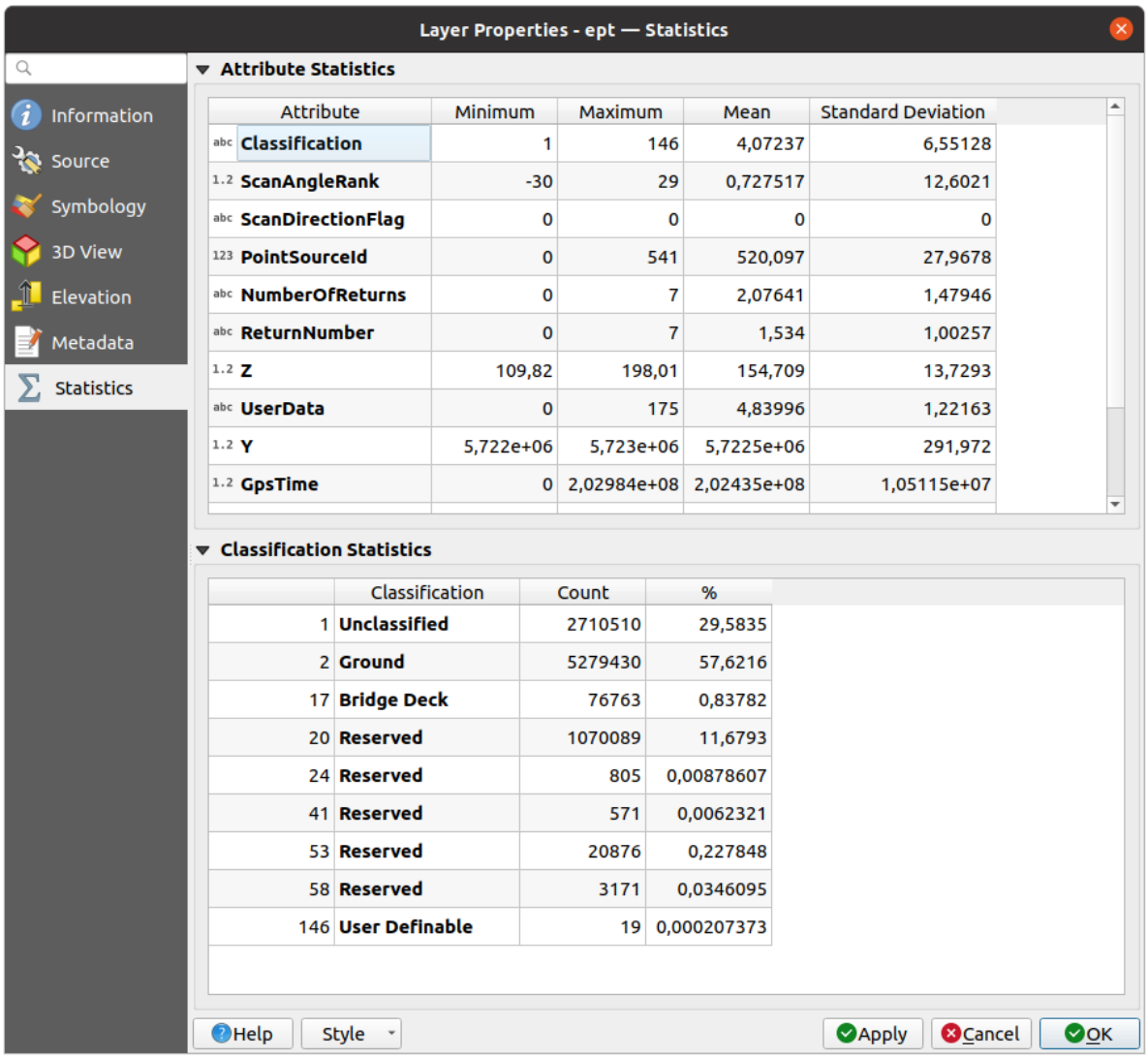


図 20.10: 点群 統計量の出力タブ

第21章　地図のレイアウト

印刷レイアウトやレポートを使用すると、地図や地図帳を作成して印刷したり、画像、PDF、また SVG ファイルとして保存できます。

21.1 印刷レイアウトの概要

印刷レイアウトは、図面のレイアウトおよび印刷機能を提供します。 QGIS マップキャンバス、テキストラベル、画像、凡例、スケールバー、基本的な図形、矢印、属性テーブル、HTML フレームなどの要素を追加できます。各要素のサイズ変更、グループ化、整列、配置、回転を行い、プロパティを調整してレイアウトを作成できます。レイアウトは印刷したり、画像形式や PostScript、PDF、SVG にエクスポートしたりできます。レイアウトはテンプレートとして保存し、別のセッションで再度読み込むことができます。最後に、テンプレートに基づいて複数の地図を生成するには、地図帳の作成を使用します。

21.1.1 初心者向けサンプルセッション

印刷レイアウトで作業を始める前に、QGIS のマップキャンバスにいくつかのラスタレイヤまたはベクタレイヤを読み込み、それらのプロパティを自分の使いやすいように調整する必要があります。すべてがレンダリングされ、好みのシンボロジになったら、 **プロジェクト** ツールバーの 新規印刷レイアウト アイコンをクリックするか、 **プロジェクト** **新規印刷レイアウト** を選択します。新しいレイアウトのタイトルを選択するプロンプトが表示されます。

地図の作成手順を説明するために、以下の指示に従ってください。

1. 左側で 地図を追加 ツールバーボタンを選び、左マウスボタンを押しながらキャンバスに四角形を描きます。すると、キャンバスに描いた四角形の内部に QGIS マップビューが表示されます。
2. スケールバーを追加 ツールバーボタンを選び、印刷レイアウトキャンバス上でマウスの左ボタンをクリックします。スケールバーがキャンバスに追加されます。
3. 凡例を追加 ツールバーボタンを選び、左マウスボタンを押しながらキャンバスに四角形を描きます。描いた四角形の内部に凡例が描かれます。
4. アイテムを選択/移動 ボタンを選び、キャンバス上の地図を選択して、少し動かします。
5. 地図アイテムが選ばれている間は、地図アイテムのサイズも変更できます。地図アイテムのコーナーの小さな白い四角形の 1 つで左マウスボタンを押し、新しい位置までドラッグしてサイズを変更します。

6. 左下の **アイテムプロパティ** パネルをクリックし、地図の回転の設定を探します。 **地図の回転** を '15.00 ° ' に変更してください。地図アイテムの向きが変わることが確認できます。

7. これで、 **レイアウト** メニューのエクスポートツールを使用して、印刷レイアウトを印刷したり、画像形式、PDF、または SVG にエクスポートできます。

8. 最後に、 プロジェクトを保存 ボタンを使用して、この印刷レイアウトをプロジェクトファイル内に保存できます。

印刷レイアウトには複数の要素を追加できます。印刷レイアウトキャンバスには、 1 つまたは複数のページに、複数の地図ビュー、凡例、またはスケールバーを配置することもできます。各要素には独自のプロパティがあり、地図の場合は独自の範囲があります。レイアウトキャンバスから要素を削除する場合は、`Delete` キーまたは `Backspace` キーを使用して削除できます。

21.1.2 レイアウトマネージャ

レイアウトマネージャ は、プロジェクトの印刷レイアウトを管理するメインウィンドウです。プロジェクト内の既存の印刷レイアウトとレポートの概要を提供し、以下のツールがあります：

- レイアウトの検索
- 空からまたはテンプレートを使用して、新しい印刷レイアウトまたは新しいレポートを追加したり、既存のものを複製する
- レイアウトやレポートの名前の変更や削除
- プロジェクトでレイアウトやレポートを開いて表示

レイアウトマネージャダイアログを開くには：

- メイン QGIS ダイアログから **プロジェクト** → **レイアウトマネージャ**... メニューを選択するか、 **プロジェクトツールバー** の レイアウトマネージャの表示 ボタンをクリックします。
- 印刷レイアウトまたはレポートダイアログから、 **レイアウト** → **レイアウトマネージャ**... メニューを選択するか、 **レイアウトツールバー** の レイアウトマネージャ ボタンをクリックします。

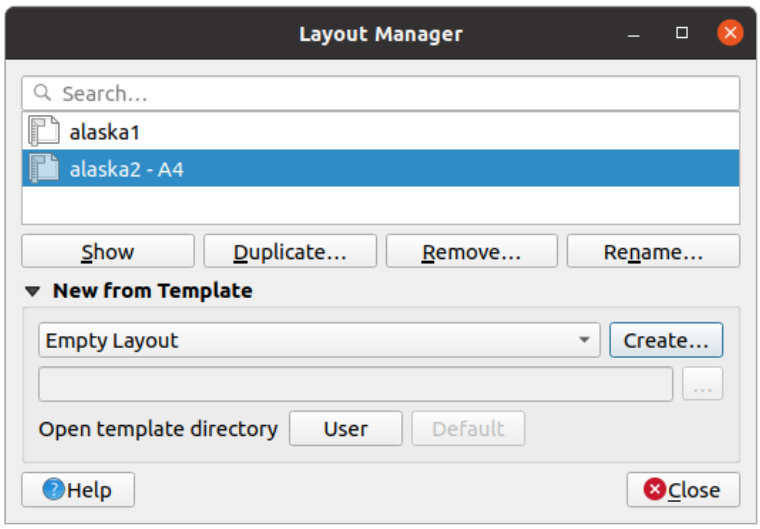


図 21.1: 印刷レイアウトマネージャ

レイアウトマネージャの上部には、プロジェクトで使用可能なすべての印刷レイアウトまたはレポートが一覧表示されます。これには、以下のツールがあります：

- 選択を表示：複数のレポートや印刷レイアウトを選択し、ワンクリックでそれらを開くことができます。名前をダブルクリックしても開きます。
- 選択した印刷レイアウトまたはレポートの複製（アイテムが 1 つ選択されている場合のみ使用可能）：選択した印刷レイアウトまたはレポートをテンプレートとして、新しいダイアログを作成します。新しいレイアウトのタイトルを入力するようプロンプトが表示されます。
- レポートやレイアウトの名前の変更（アイテムが 1 つ選択されている場合のみ使用可能）：レイアウトの新しいタイトルを入力するプロンプトが表示されます。
- レイアウトの削除：選択した印刷レイアウト（複数可） をプロジェクトから削除します。

レイアウトマネージャの下部のツールを使えば、新しい印刷レイアウトまたはレポートを空から作成することも、テンプレートから作成することもできます。QGIS はデフォルトで、ユーザプロファイルとアプリケーションテンプレートのディレクトリ（フレームの下部にある 2 つのボタンでアクセス可能）内のテンプレートを検索しますが、 **設定** **オプション** **レイアウト** で **追加のプリントテンプレートを探すパス** として宣言されたフォルダも検索します。見つかったテンプレートはコンボボックスにリストされます。アイテムを選択し **作成** ボタンを押すと、新しいレポートまたは印刷レイアウトを作成します。

カスタムフォルダからレイアウトテンプレートを使用することもできます。その場合、テンプレートのドロップダウンリストで **テンプレートを指定** を選択し、テンプレートを参照して **作成** を押します。

Tip: ブラウザパネルからテンプレートに基づく印刷レイアウトを作成する

印刷レイアウトテンプレート `.qpt` ファイルを任意のファイルブラウザから地図キャンバスにドラッグ＆ドロップするか、 **ブラウザパネル** でダブルクリックすると、テンプレートから新しい印刷レイアウトが作成されます。

21.1.3 印刷レイアウトのメニュー、ツール、パネル

印刷レイアウトを開くと、印刷オプションを使用するときの紙面を表す空白のキャンバスが表示されます。初期状態ではキャンバスの左側に、現在の QGIS マップキャンバスやテキストラベル、画像、凡例、スケールバー、基本的な図形、矢印、属性テーブル、HTML フレームといった印刷レイアウトアイテムを追加するためのボタンがあります。また、このツールバーには、ナビゲーション、エリアのズームイン、レイアウトビューのパンを行うためのボタンがあるほか、レイアウトアイテムの選択や、地図アイテムのコンテンツを移動するためのボタンもあります。

図 21.2 に、要素を追加する前の初期状態の印刷レイアウトを示します。

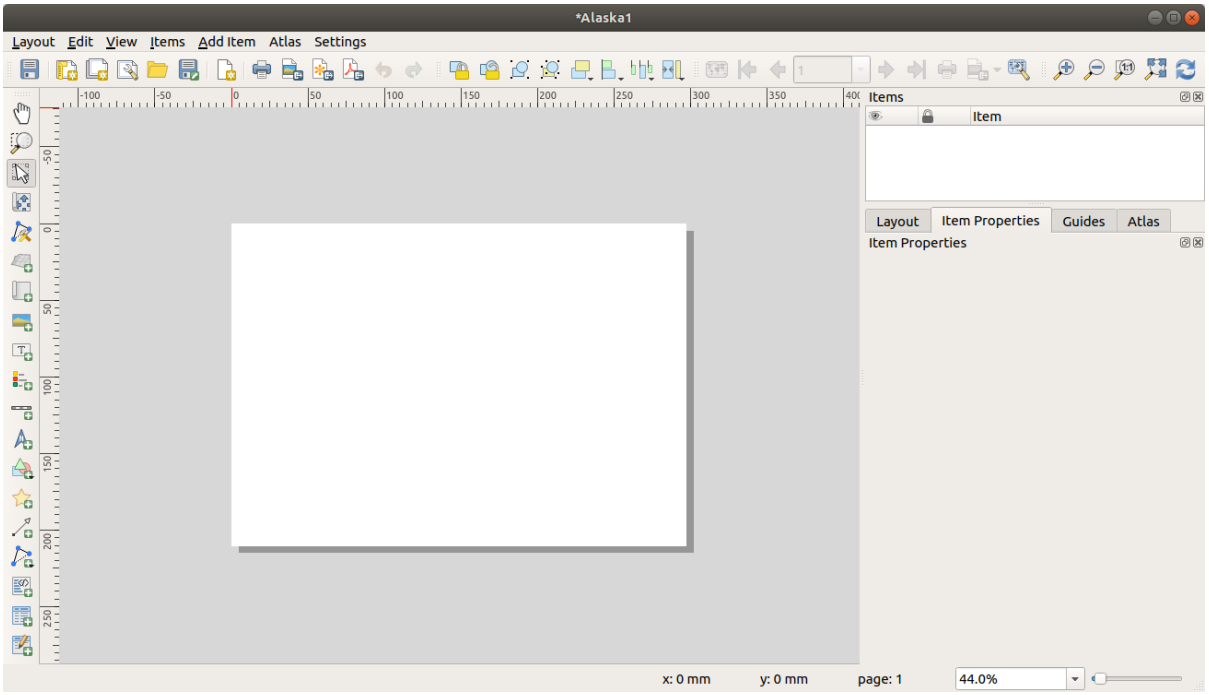


図 21.2: 印刷レイアウト

キャンバスの右側に 2 組のパネルがあります。上のパネルには **アイテム** と **編集履歴** のパネル があり、下のパネルには **レイアウト** 、 **アイテムプロパティ** 、 **地図帳** のパネルがあります。

- **アイテム** パネルは、キャンバスに追加されたすべての印刷レイアウトアイテムのリストと、それらをグローバルに操作する方法を提供します（詳細は **アイテムパネル** 参照）。
- **編集履歴** パネルには、レイアウトに適用したすべての変更の履歴が表示されます。マウスをクリックすると、レイアウトの編集ステップを元に戻したりやり直したりして、どこかの状態に戻すことができます。
- **レイアウト** パネルは、エクスポートまたは内部で作業するときにレイアウトに適用する一般的なパラメータを設定できます（詳細は **レイアウトパネル** 参照）。
- **アイテムプロパティ** パネルには、選択したアイテムのプロパティが表示されます。 キャンバス上のアイテム（凡例、スケールバー、ラベルなど）を選択するには、 アイテムを選択/移動 アイコンを使用します。次に、 **アイテムプロパティ** パネルをクリックし、選択したアイテムの設定をカスタマイズします（それぞれのアイテム設定の詳細については **レイアウトアイテム** 参照）。
- **地図帳** パネルを使用すると、現在のレイアウトに対する地図帳の作成を有効にし、そのパラメータにアクセスできます（地図帳作成の使用法の詳細については **地図帳の作成** を参照）。

印刷レイアウトウィンドウの下部にはステータスバーがあり、マウスの位置、現在のページ番号、ズームレベルを設定するコンボボックス、該当する場合は選択されたアイテム数、地図帳作成の場合は地物の番号が表示されます。

印刷レイアウトウィンドウの上部には、メニューやその他のツールバーがあります。すべての印刷レイアウトツールは、メニューおよびツールバーのアイコンから使用できます。

ツールバーとパネルは、任意のツールバー上でマウスの右ボタンを使用するか、 **ビュー** **ツールバー** または **ビュー** **パネル** で表示非表示を切り替えできます。

メニューとツール

レイアウトメニュー

レイアウト では、レイアウトを管理する操作を提供します：

- 印刷レイアウトウィンドウからプロジェクトファイルを直接保存します。
-  **新規レイアウト**... を使用して、新しい空の印刷レイアウトを作成します。
-  **レイアウトの複製**...：現在の印刷レイアウトを複製して、新しい印刷レイアウトを作成します。
-  **レイアウトの削除**... で現在のレイアウトを削除します。
-  **レイアウトマネージャ**... を開きます。
- **レイアウト**　：既存の印刷レイアウトを開きます。

レイアウトのデザインができたら、  **テンプレートとして保存** や  **テンプレートからアイテムを追加** のアイコンを使用すると、現在の印刷レイアウトセッションの状態を `.qpt` テンプレートファイルとして保存したり、そのアイテムを再度、別のセッションや印刷レイアウトに読み込むことができます。

レイアウト メニューには、QGIS で作成した地理情報をレポートに取り入れたり、公開したりして共有するための強力な方法があります。それは、  **画像としてエクスポート**... 、  *PDF* **としてエクスポート**... 、  *SVG* **としてエクスポート**... 、そして  **印刷**... といったツールです。

以下の表は、このメニューで利用可能なすべてのツールと、便利な情報のリストです。

ツール	ショートカット	ツールバー	参照
プロジェクトを保存	Ctrl+S	**レイアウト**	*QGIS* **プロジェクトの紹介**
新規印刷レイアウト	Ctrl+N	**レイアウト**	**レイアウトマネージャ**
レイアウトの複製		**レイアウト**	**レイアウトマネージャ**
レイアウトの削除			
レイアウトマネージャ...		**レイアウト**	**レイアウトマネージャ**
レイアウト			
レイアウトのプロパティ...			**レイアウトパネル**
レイアウト名の変更...			
ページを追加...		**レイアウト**	**ページのプロパティの操作**
テンプレートからアイテムを追加		**レイアウト**	**レイアウトアイテムの作成**
テンプレートとして保存...		**レイアウト**	**レイアウトマネージャ**
画像としてエクスポート...		**レイアウト**	**画像としてエクスポート**
SVG **としてエクスポート...**		**レイアウト**	*SVG* **としてエクスポート**
PDF **としてエクスポート...**		**レイアウト**	*PDF* **としてエクスポート**
ページ設定...	Ctrl+Shift+P		
印刷...	Ctrl+P	**レイアウト**	**出力の作成**
閉じる	Ctrl+Q		

編集メニュー

編集 メニューには、印刷レイアウトのアイテムを操作するツールがあります。これには、レイアウト内のアイテムの選択ツール、コピー／切り取り／貼り付け、元に戻す/やり直す（**編集履歴パネル：操作の取り消しと再実行** 参照）機能などの一般的な操作が含まれます。

貼り付けアクションを使用すると、要素は現在のマウス位置に貼り付けられます。 **編集**　**領域に貼り付け** アクションを使用するか、 Ctrl+Shift+V を押すと、元々アイテムがあったページと同じ位置で、現在のページにアイテムを貼り付けます。これにより、ページ間で同じ位置にアイテムをコピー／貼り付けできます。

以下の表は、このメニューで利用可能なすべてのツールと、便利な情報のリストです。

表 21.1: 利用可能なツール

ツール	ショートカット	ツールバー	参照
（最後の変更を）元に戻す	Ctrl+Z	レイアウト	編集履歴パネル：操作の取り消しと再実行
（最後に取り消した変更を）やり直す	Ctrl+Y	レイアウト	編集履歴パネル：操作の取り消しと再実行
削除	Del		
切り取り	Ctrl+X		
コピー	Ctrl+C		
貼り付け	Ctrl+V		
領域に貼り付け	Ctrl+Shift+V		
すべてを選択	Ctrl+A		
すべてを選択解除	Ctrl+Shift+A		
選択範囲を反転			
下のアイテムを選択	Ctrl+Alt+[		
上のアイテムを選択	Ctrl+Alt+]		
レイアウトのパン	P	ツールボックス	
ズーム	Z	ツールボックス	
アイテムの選択／移動	V	ツールボックス	レイアウトアイテムの操作
コンテンツの移動	C	ツールボックス	地図アイテム
ノードアイテムの編集		ツールボックス	ノードに基づく図形アイテム

ビューメニュー

ビュー メニューはナビゲーションツールへのアクセスを提供し、印刷レイアウトの一般的な動作を設定するのに役立ちます。一般的なズームツールのほか、以下の機能があります：

- リフレッシュ （ビューの状態が地図と不整合である場合に使用してください）
- **グリッド** を有効にすると、アイテムを移動したり作成したりする際に、グリッドにスナップさせることができます。グリッドの設定は、 **設定** **レイアウトオプション**... または **レイアウトパネル** で行えます。
- **ガイド** を有効にすると、アイテムを移動したり作成したりする際に、ガイドにアイテムをスナップさせることができます。ガイドはルーラー（レイアウトの上部または左側）をクリックし目的の場所

にドラッグ＆ドロップすることで作成でき、赤い線で表示されます。

- **スマートガイド** ：他のレイアウトアイテムをガイドとして使用し、アイテムを移動または変形するときに、これに動的にスナップさせる機能です。
- **ガイドのクリア** 現在のすべてのガイドを削除します。
- **バウンディングボックスを表示する** ：バウンディングボックスをアイテムの周りに表示することで、選択するときに識別しやすくなります。
- **定規の表示** レイアウトの周囲にルーラーを表示します。
- **ページの表示** ：チェックを外すとページを透明に設定します。レイアウトは、プレゼンテーションや他のドキュメントに含めるなど、非印刷用のレイアウトを作成するために使用することがしばしばあります。このためには、完全に透明な背景を使用してコンポジションをエクスポートすることが望まれます。他の編集パッケージでは、「無限キャンバス」と呼ばれることもあります。

印刷レイアウトでは、マウスホイールやステータスバーのスライダー、コンボボックスを使用してズームレベルを変更できます。レイアウトエリアで作業中にパンモードに切り替える必要がある場合は、Spacebar またはマウスホイールを押したままにします。Ctrl+Spacebar でズームインモードに、Ctrl+Alt+Spacebar でズームアウトモードに一時的に切り替えできます。

パネルとツールバーは、**ビュー** メニューから有効にできます。コンポジションの操作スペースを最大化するには、**ビュー** **パネルの表示切り替え** オプションをチェックするか、Ctrl+Tab を押します。するとすべてのパネルが非表示となり、チェックを外すと、以前に表示されていたパネルだけが復元されます。

また、F11 を押すか **ビュー** **フルスクリーン切り替え** を使用することで、全画面モードに切り替えて操作スペースを増やせます。

ツール	ショートカット	ツールバー	参照
リフレッシュ	F5	ナビゲーション	
プレビュー			
拡大	Ctrl++	ナビゲーション	
縮小	Ctrl+-	ナビゲーション	
*100%*にズーム	Ctrl+1	ナビゲーション	
全域表示	Ctrl+0	ナビゲーション	
ページ幅にズーム			
グリッドを表示	Ctrl+'		ガイドとグリッド
グリッドにスナップ	Ctrl+Shift+'		ガイドとグリッド
ガイドの表示	Ctrl+;		ガイドとグリッド
ガイドにスナップ	Ctrl+Shift+;		ガイドとグリッド
スマートガイド	Ctrl+Alt+;		
ガイドを管理する...			ガイドパネル
ガイドのクリア			ガイドパネル
定規の表示	Ctrl+R		
バウンディングボックスを表示する	Ctrl+Shift+B		
ページの表示			
ツールバー			パネルとツールバー
パネル			パネルとツールバー
フルスクリーン切り替え	F11		ビュー
パネル表示切り替え	Ctrl+Tab		ビュー

アイテムメニュー

アイテム は、レイアウト内のアイテムの位置やアイテム間の関係を設定するのに役立ちます（**レイアウトアイテムの操作** を参照）。

ツール	ショートカット	ツールバー	参照
グループ化	Ctrl+G	アクション	アイテムのグループ化
グループ解除	Ctrl+Shift+G	アクション	アイテムのグループ化
上へ	Ctrl+]	アクション	整列と等間隔に並べる
下へ	Ctrl+[	アクション	整列と等間隔に並べる
最前面に	Ctrl+Shift+]	アクション	整列と等間隔に並べる
最背面に	Ctrl+Shift+[	アクション	整列と等間隔に並べる
選択アイテムを固定する	Ctrl+L	アクション	アイテムのロック
すべてアンロックする	Ctrl+Shift+L	アクション	アイテムのロック
整列		アクション	整列と等間隔に並べる
均等配置		アクション	アイテムの移動とサイズ変更
サイズの変更		アクション	アイテムの移動とサイズ変更

追加メニュー

これらは、レイアウトアイテムを作成するためのツールです。各ツールは **レイアウトアイテム** の章で詳しく説明されています。

ツール	ツールバー	参照
地図を追加	ツールボックス	地図アイテム
3D 地図を追加	ツールボックス	*3D* 地図アイテム
画像を追加	ツールボックス	画像アイテム
ラベルを追加	ツールボックス	ラベルアイテム
動的テキストを追加		ラベルアイテム
凡例を追加	ツールボックス	凡例アイテム
スケールバーを追加	ツールボックス	スケールバーアイテム
方位記号を追加	ツールボックス	方位記号アイテム
Shape を追加	ツールボックス	基本的な *Shape* アイテム
四角形を追加	ツールボックス	基本的な *Shape* アイテム
楕円を追加	ツールボックス	基本的な *Shape* アイテム
三角形を追加	ツールボックス	基本的な *Shape* アイテム
マーカーを追加	ツールボックス	マーカーアイテム
矢印を追加	ツールボックス	矢印アイテム
ノードアイテムを追加	ツールボックス	ノードに基づく図形アイテム
ポリゴン（*Polygon*）を追加	ツールボックス	ノードに基づく図形アイテム
ポリラインを追加	ツールボックス	ノードに基づく図形アイテム
HTML を追加	ツールボックス	*HTML* フレームアイテム
属性テーブルを追加	ツールボックス	属性テーブルアイテム
固定テーブルを追加	ツールボックス	固定テーブルアイテム

地図帳メニュー

ツール	ショートカット	ツールバー	参照
地図帳のプレビュー	Ctrl+ALt+/	**地図帳**	地図帳のプレビューと作成
最初の地物	Ctrl+<	**地図帳**	地図帳のプレビューと作成
前の地物	Ctrl+,	**地図帳**	地図帳のプレビューと作成
次の地物	Ctrl+.	**地図帳**	地図帳のプレビューと作成
最後の地物	Ctrl+>	**地図帳**	地図帳のプレビューと作成
地図帳の印刷...		**地図帳**	地図帳のプレビューと作成
画像として地図帳をエクスポート...		**地図帳**	地図帳のプレビューと作成
SVG **として地図帳をエクスポート...**		**地図帳**	地図帳のプレビューと作成
PDF **として地図帳をエクスポート...**		**地図帳**	地図帳のプレビューと作成
地図帳の設定		**地図帳**	地図帳の作成

設定メニュー

設定 **レイアウトオプション...** メニューは、QGIS メインキャンバスの **設定** **オプション** **レイアウト** メニューへのショートカットです。ここでは、新しい印刷レイアウトでデフォルトとして使用されるいくつかのオプションを設定できます。

- **レイアウトのデフォルト** では、使用するデフォルトのフォントを指定できます。
- **グリッドの外観** では、グリッドのスタイルと色を設定できます。グリッドには、 **点** 、 **塗りつぶし線** 、 **十字** の 3 種類があります。
- **グリッドとガイドのデフォルト** では、グリッドの間隔、オフセット、スナップ許容値を定義します（詳細は **ガイドとグリッド** 参照）。
- **レイアウトのパス** ：プリントテンプレートを検索するためのカスタムパスのリストを管理します。

コンテキストメニュー

印刷レイアウトダイアログで右クリックした場所に応じて、さまざまな機能を備えたコンテキストメニューを開きます。

- メニューバーまたは任意のツールバーを右クリックすると、レイアウトパネルとツールバーのリストが表示され、ワンクリックで有効または無効にできます。
- 定規を右クリックすると、 **ガイドの表示** 、 **ガイドにスナップ** 、 **ガイドパネル** を開く **ガイドを管理する...** 、 **ガイドのクリア** の操作ができます。また、定規を隠すこともできます。
- 印刷レイアウトキャンバスを右クリックすると、以下の操作をができます：

- 最近の変更を **元に戻す** 、 **やり直す** 、またはコピーしたアイテムの **貼り付け** ができます（アイテムが選択されていない場合のみ使用可能）。
- ページ上をクリックすると、上記に加えて現在の **ページのプロパティ** パネルや、 **ページを削除** にもアクセスできます。
- 選択したアイテムを右クリックした場合には、そのアイテムの切り取りやコピーができるほか、**アイテムのプロパティ** パネルを開くことができます。
- 複数のアイテムが選択されている場合にはそれらをグループ化したり、少なくとも 1 つのグループがすでに選択されている場合にはグループ化を解除したりできます。

- レイアウトパネルのテキストボックスやスピンボックスウィジェット内を右クリックすると、そのコンテンツを操作するための編集オプションが提供されます。

レイアウトパネル

レイアウト パネルでは、印刷レイアウトのグローバル設定を定義できます。

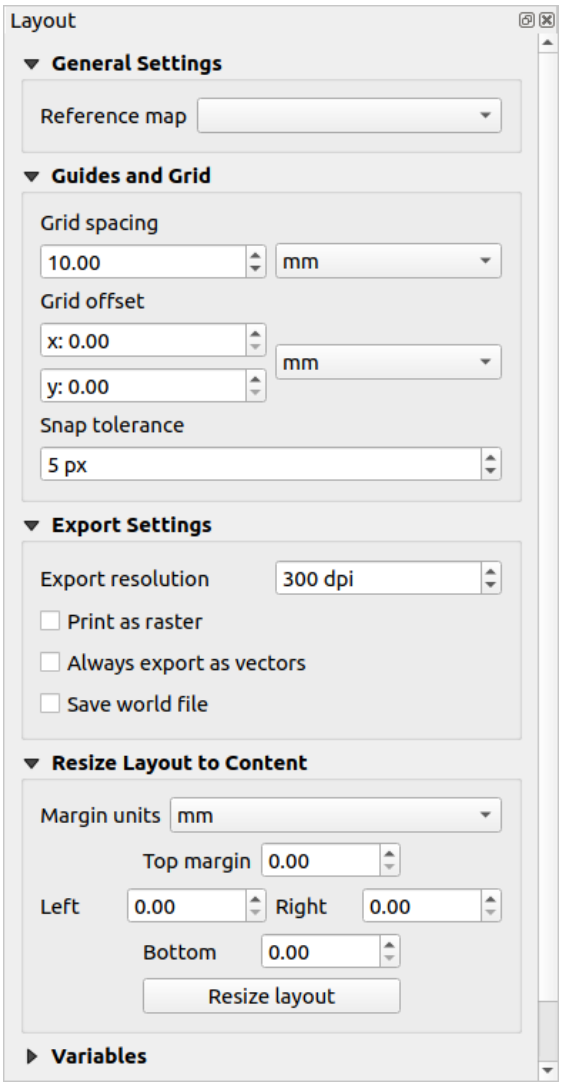


図 21.3: 印刷レイアウトのレイアウト設定

一般設定

印刷レイアウトでは、複数の地図アイテムが使用できます。 **参照マップ** とは、レイアウトのマスター地図として利用する地図アイテムのことです。レイアウト内に地図アイテムがある限りは、参照マップが割り当てられます。レイアウトは、プロパティや変数で単位や縮尺を計算する際にこの参照マップを使用します。これには、印刷レイアウトをジオリファレンスされた形式にエクスポートすることも含まれます。

さらに、スケールバーや凡例、方位記号などの新しいレイアウトアイテムは、デフォルトでは描画したアイテムの下にある地図アイテムに設定（方向、表示レイヤ、縮尺など）がバインドされますが、これらと地図アイテムに重なりが無い場合には、参照マップの設定が使われます。

ガイドとグリッド

アイテムを正確に配置するために、ページにいくつかの参照マークを付けることができます。これらのマークには、次のものがあります：

- 必要な位置に配置した、(**ガイド** と呼ばれる）シンプルな水平線または垂直線（ガイドの作成については **ガイドパネル** 参照)。
- 規則的な **グリッド** ：レイアウト上に重ねられた水平線と垂直線の網目。

グリッド間隔 や **グリッドオフセット** などの設定は、アイテムに使用する **スナップ許容量** と同様に、このグループで調整できます。スナップ許容量とは、アイテムの移動、サイズ変更、または作成中にマウスカーソルがグリッドまたはガイドにスナップする最大距離です。

グリッドやガイドを表示するかは、 **ビュー** メニューで設定します。そこでは、これらをレイアウトアイテムのスナップに使用するかどうかも決定できます。グリッド線とガイド線の両方がポイントの許容範囲内にある場合には、常にガイドが優先されます。なぜなら、ガイドは手動で設定されているため（したがって、それらは非常に望ましいスナップ位置に明示的に配置されており、一般的なグリッドよりも優先して選択されるべきであるという前提）です。

注釈: 設定 **レイアウトオプション** メニューでも、上記のグリッドとガイドのパラメータを設定することができます。ただし、これらのオプションは新しい印刷レイアウトにのみデフォルトとして適用されます。

エクスポート設定

エクスポートするすべての地図に使用する解像度を **エクスポート解像度** で定義できます。この設定は、地図をエクスポートするたびにオーバーライドできます。

いくつかの高度なレンダリングオプション（ **ブレンドモード** や **描画エフェクト** 等）のために、レイアウトアイテムを正しくエクスポートするためにはラスタ化が必要な場合があります。 QGIS は、他のすべてのアイテムを強制的にラスタ化することなく、必要なものを個別にラスタ化します。これにより、PostScript や PDF として印刷または保存する際に、可能な限りベクタとしてアイテムを保持できます。例えば、レイヤに不透明度を持つ地図アイテムがあっても、強制的にラベルやスケールバーなどもラスタ化されることはありません。ただし、以下の設定もできます：

- **ラスタとして印刷する** のボックスにチェックを入れると、すべてのアイテムを強制的にラスタ化します。
- これとは正反対のオプション **常にベクタとしてエクスポートする** を使用すると、ベクタに互換性のある形式にエクスポートしたときに、アイテムをベクタとして保持するように強制します。ただし、場合によっては、これにより出力がレイアウトと異なる見た目となる可能性があることに注意してください。

可能な形式（例： `.TIF`、`.PDF`）の場合には、印刷レイアウトをエクスポートすると、デフォルトで（**一般設定** グループの **参照マップ** アイテムに基づいて）ジオリファレンスされたファイルが作成されます。他の形式の場合には、ジオリファレンスされた出力の作成には、**ワールドファイルを保存する** にチェックを入れてワールドファイルを生成する必要があります。このワールドファイルはエクスポートされたマップと同じ場所に作成され、参照マップアイテムを含むページ出力の名前を持ち、簡単にジオリファレンスするための情報を含みます。

レイアウトの大きさを内容に合わせる

このグループの **レイアウトのリサイズ** ツールを使用すると、印刷レイアウトの現在のコンテンツをカバーする範囲の、固有なページ構成を作成します（トリミングされた境界の周囲には、追加で **マージン** を持ちます）。

この動作は、 **内容に合わせて切り取る** オプションとは異なることに注意してください。既存のすべてのページの代わりに、すべてのアイテムが実際に唯一のページに配置されます。

変数

変数 には、レイアウトのレベルで使用可能なすべての変数がリストされます（すべてのグローバル変数とプロジェクト変数が含まれます）。

また、ここではレイアウトレベルの変数も管理できます。 ボタンをクリックすると、新しいレイアウトレベルのカスタム変数を追加できます。同様に、リストからレイアウトレベルのカスタム変数を選択して ボタンを押すと、変数の削除ができます。

変数の使用方法の詳細は **一般ツール** セクションにあります。

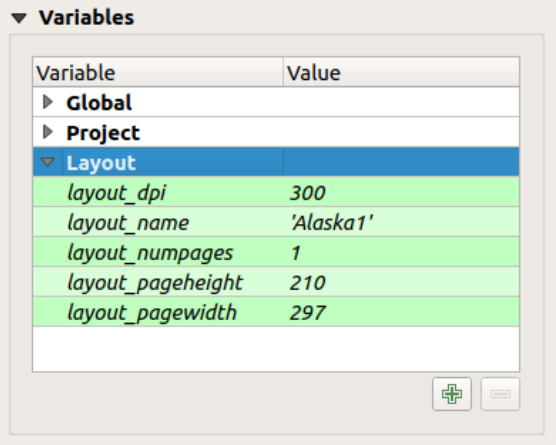


図 21.4: 印刷レイアウトの変数エディタ

ページのプロパティの操作

レイアウトは複数のページで構成できます。たとえば、最初のページにはマップキャンバスを、2 番目のページにはレイヤに関連付けられた属性テーブルを、3 番目のページには組織の Web サイトにリンクする HTML フレームを表示できます。また、各ページに多くの種類のアイテムを追加できます。

新しいページの追加

さらに、異なるサイズやページの向きを使用してレイアウトを作成できます。ページを追加するには、 **レイアウト** メニューまたは **レイアウトツールバー** から **ページを追加**... ツールを選択します。 **ページの挿入** ダイアログが開き、以下の情報の入力を求められます：

- 挿入するページ数
- ページの位置：特定のページの前または後、または印刷レイアウトの最後。
- **ページサイズ** ：事前に設定された形式のページ（ `A4` 、 `B0` 、 `Legal` 、 `Letter` 、 `ANSI A` 、`Arch A` およびその派生形式、または `1920x1080` や `1024x768` などの解像度タイプと、その **向き** （縦または横）。

  ページサイズは カスタム 形式にすることもできます。その場合、 **幅** と **高さ** を（必要に応じてロックされたサイズ比で）入力し、使用する単位を `mm` 、 `cm` 、 `px` 、 `pt` 、 `in` 、 フィート などから選択する必要があります。ある単位から別の単位に切り替えるときには、入力された値の変換が自動的に適用されます。

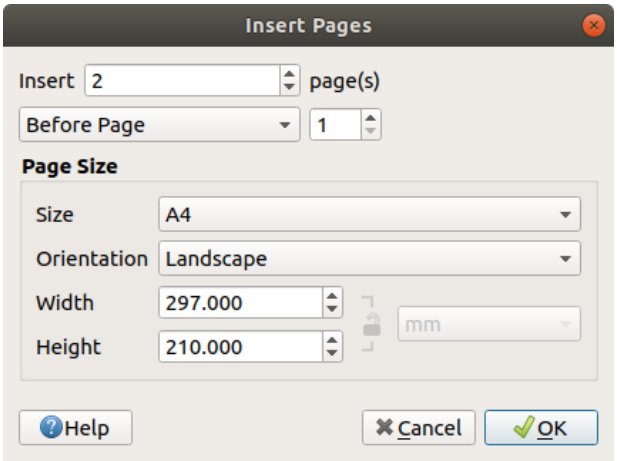


図 21.5: 印刷レイアウトに新しいページを作成する

ページのプロパティの更新

ページの **アイテムプロパティ** パネルを使用して、後から任意のページをカスタマイズできます。ページのプロパティにアクセスするには、ページの空白部分を左クリックするか、ページ上を右クリックしてから **ページのプロパティ...** を選択します。 **アイテムプロパティ** パネルが開き、次のような設定が表示されます：

- 上記の **ページサイズ** フレーム。データによって定義された上書きオプションを使用して、各プロパティを変更できます（使用法については **地図帳でのデータによって定義された上書きの利用** 参照）。
- **ページをエクスポートから除外する** は、現在のページとそのコンテンツを **レイアウト出力** に含めるかどうかを制御します。
- 現在のページの **背景** は、好みの **色** や **シンボル** を使用して設定できます。

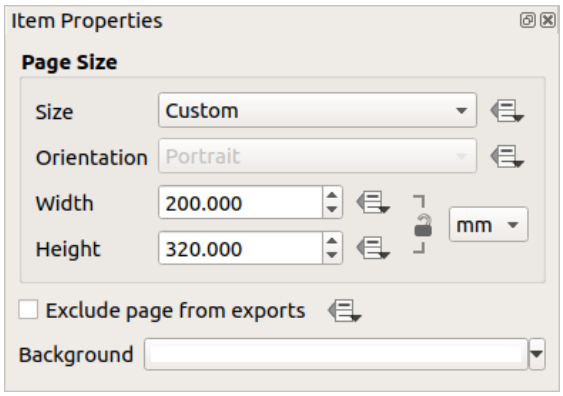


図 21.6: ページのプロパティダイアログ

ガイドパネル

ガイドとは、アイテムの作成、移動、サイズ変更などの際に、アイテムの配置を補助するためにレイアウトページ上に配置できる垂直または水平の参照線です。ガイドを有効にするには、 **ビュー　ガイドの表示** と **ビュー　ガイドにスナップ** オプションにチェックを入れる必要があります。ガイドを作成するには、2種類の方法があります：

- **ビュー　定規の表示** オプションがオンになっている場合には、定規をドラッグしてきてページ内の任意の位置でマウスボタンを離すと、ガイドが作成されます。
- さらに精度を求めるなら、 **ビュー　ツールボックス** の **ガイド** パネルを利用するか、ページのコンテキストメニューにある **ページのガイドを管理...** を選択して下さい。

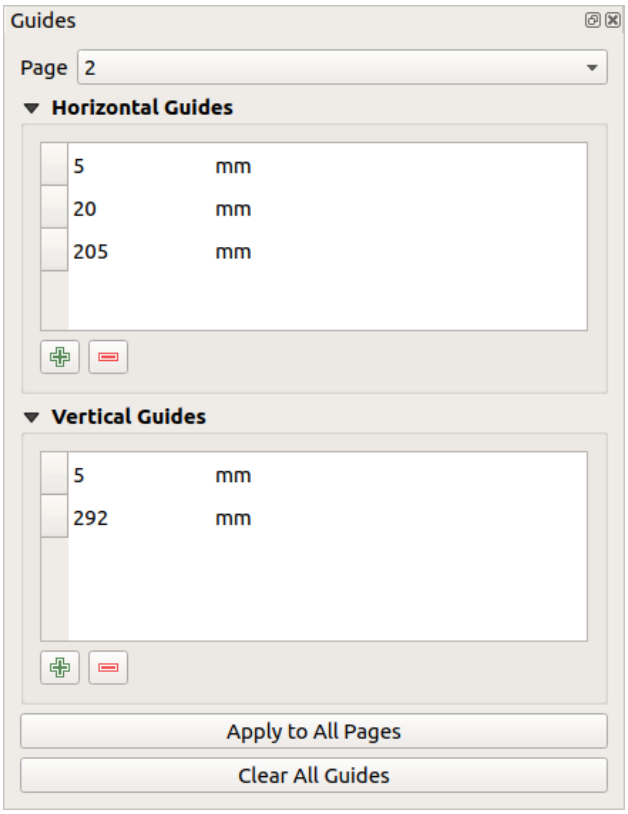


図 21.7: ガイドパネル

ガイド パネルでは、以下の手順で指定した場所にスナップラインを作ることができます。

1. ガイドを追加したい **ページ** を選択して下さい。
2. 新しいガイドを追加 ボタンをクリックして、水平または垂直の座標を入力して下さい。原点は左上の角です。異なる距離単位を使用することもできます。

   また、このパネルでは既存のガイドを正確な座標に調整することもできます。値をダブルクリックして変更して下さい。
3. **ガイド** パネルには、現在のページのガイドのみがリストされます。このため、現在のページのガイドのみを作成・削除できます。ただし、 **すべてのページに適用** ボタンを使うと、現在のページのガイド設定を他のページにも複製することができます。
4. ガイドを削除するには、削除したいガイドを選択して 選択したガイドを削除 ボタンを押します。 **すべてのガイドをクリア** ボタンを押すと、現在のページの全てのガイドを削除します。

Tip: 既存のレイアウトアイテムにスナップする

ガイドとグリッド以外にも、アイテムの移動やサイズ変更、新規作成の際に、既存のアイテムをスナップ参照として使用できます。これは **スマートガイド** と呼ばれ、 **ビュー** **スマートガイド** オプションがチェックされている必要があります。マウスポインタがアイテムの境界に近づくと、スナップクロスが表示されます。

アイテムパネル

アイテム パネルには、アイテムの選択や表示を管理するためのいくつかのオプションがあります。印刷レイアウトキャンバスに追加されたすべてのアイテム（ **アイテムグループ** を含む）がリストに表示されます。アイテムを選択すると、リストで対応する行が選択され、リストの行を選択すると、印刷レイアウトキャンバス中の対応するアイテムが選択されます。これは、他のアイテムの背後にあるアイテムを選択する便利な方法です。選択された行は太字で表示されます。

選択されたアイテムに対して、以下の操作ができます：

- アイテムの表示・非表示の設定
- 位置のロック・アンロック
- Z 位置の並べかえ。リスト内の各アイテムをクリック＆ドラッグで上下に移動できます。リストの上位のアイテムは、印刷レイアウトのキャンバスの前面に表示されます。デフォルトでは、新しく作成されたアイテムは最前面に配置されます。
- テキストをダブルクリックすることで、アイテム ID を変更します。
- アイテムを右クリックすると、アイテムのコピーや削除、 **プロパティパネル** を開くことができます。

アイテムが正しい位置に配置されたら、 列のボックスをチェックして位置をロックできます。ロックされたアイテムはキャンバスで選択 **できません** 。ロックされたアイテムのロック解除には、 **アイテム** パネルで選んでチェックボックスのチェックを外すか、ツールバーのアイコンを使用してください。

編集履歴パネル：操作の取り消しと再実行

レイアウトの作業中に変更を取り消したり、再実行することができます。これは、 **編集** メニューや **レイアウト** ツールバーで利用可能な元に戻す・やり直すツールや、印刷レイアウト領域で右クリックしたときのコンテキストメニューで実行できます。

- 最後の変更を元に戻す
- 最後の変更をやり直す

この操作は、 **編集履歴** パネル内をマウスでクリックすることでも実行できます（ 図 21.8 を参照）。編集履歴パネルには、印刷レイアウト内で行われた最近の操作が一覧表示されます。操作を元に戻したい時点を選択し、次に新しい操作を行うと、選択した時点以降の操作は削除されます。

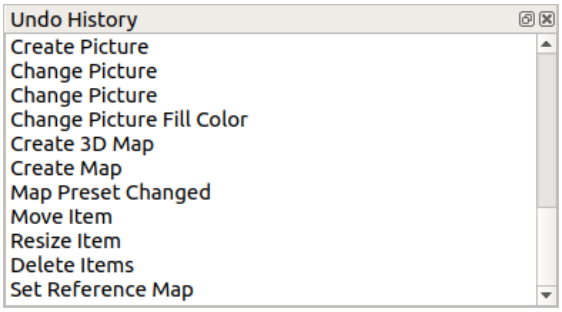


図 21.8: 印刷レイアウトの編集履歴

21.2 レイアウトアイテム

21.2.1 レイアウトアイテムの共通オプション

QGIS では、地図をレイアウトするための多数のアイテムセットを提供しています。これには、地図、凡例、スケールバー、画像、テーブル、方位記号、画像などのタイプがあります。ただし、これらのアイテムは以下に示すようにいくつかの共通なオプションと動作があります。

レイアウトアイテムの作成

レイアウトアイテムは、さまざまなツールを使用して、最初からあるいは既存のアイテムを基に作成することができます。

レイアウトアイテムを最初から作成するには、

1. **追加** メニューや **ツールボックス** バーから対応するツールを選択します。
2. 続いて、
   - ページ上をクリックし、ポップアップする **新規アイテムのプロパティ** ダイアログで必要な位置とサイズに関する情報を入力します（詳細は **位置とサイズ** を参照）。

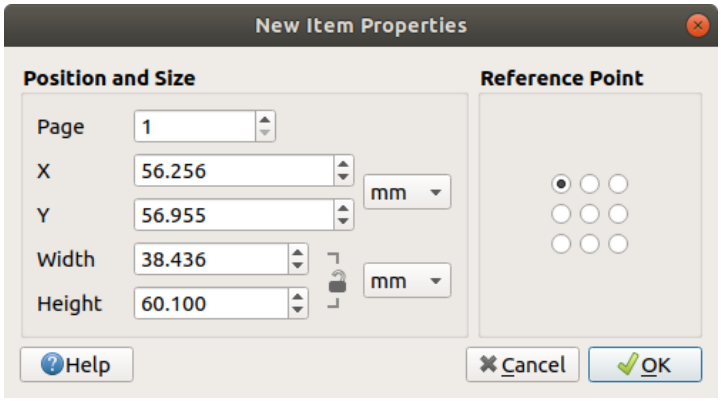


図 21.9: 新規アイテムのプロパティダイアログ

   - あるいは、クリック＆ドラッグでアイテムの初期サイズと位置を定義します。 **グリッドやガイド** のスナップを頼りにして、より良い位置に配置することができます。

注釈: ノードや矢印アイテムは特定の形状を持つため、これらはワンクリックまたはクリック＆ドラッグの方法では描画できません。クリックしてアイテムの各ノードを配置する必要があります。詳細については、 **ノードに基づく図形アイテム** を参照してください。

また、

1. **ツールボックス** ツールバーの  アイテムを選択/移動 ボタンを使って、既存のアイテムを選択します。

2. コンテキストメニューや **編集** メニューのツールを使用して、アイテムのコピーや切り取りを行い、マウス位置に新規アイテムとして貼り付けることもできます。

   さらに、 **領域に貼り付け** （ `Ctrl+Shift+V` ）コマンドを使用して、あるページのアイテムを複製し、元のページと同じ座標で別のページに配置することもできます。

その上、 **レイアウト** ▸ **テンプレートからアイテムを追加**... コマンドを使用して、印刷レイアウトのテンプレートを使用してアイテムを作成することもできます（詳細は **レイアウトマネージャ** を参照）。

Tip: ファイルブラウザを使用したレイアウトアイテムの追加

ファイルブラウザや **ブラウザ** パネルを使用して、そこから印刷レイアウトテンプレート（ `.qpt` ファイル）を印刷レイアウトダイアログ上にドラッグ＆ドロップすると、QGIS はそのテンプレートのアイテム全てをレイアウトに自動的に追加します。

レイアウトアイテムの操作

印刷レイアウト内の各アイテムは、完璧なレイアウトを作成するために移動およびサイズ変更できます。どちらの操作でも、最初のステップは  アイテムを選択/移動 ツールを有効にして、アイテムをクリックすることです。

 アイテムを選択/移動 ボタンは、複数のアイテムを選択できます。複数のアイテム上でクリックしてドラッグするか、 `Shift` キーを押しながら選択したい各アイテムをクリックします。アイテムを選択から外すには、 `Shift` キーを押しながらアイテムをクリックします。

アイテムが選択されるたびに、選択されたアイテムの数がステータスバーに表示されます。 **編集** メニュー内には、すべてのアイテムの選択、すべての選択の解除、現在の選択を反転させるなどのアクションがあります。

アイテムの移動とサイズ変更

ビュー ▸ **バウンディングボックスを表示する** オプションがチェックされていないのでなければ、選択したアイテムの境界には四角が表示され、そのうちの一つをマウスで動かせば、対応する方向にアイテムのサイズが変更されます。サイズ変更中に `Shift` キーを押しながらだと、アスペクト比が維持されます。 `Alt` キーを押しながらだと、アイテム中心を基準にサイズが変更されます。

レイアウトアイテムを移動するには、マウスで選択し、マウス左ボタンを押しながら移動させます。移動を水平軸または垂直軸に制限する必要がある場合は、キーボードの `Shift` キーを押しながらマウスを移動させます。キーボードの 矢印キー を使用しても、選択したアイテムを移動させることができます。動きが遅すぎる場合には、 `Shift` を押しながらだと移動速度を上げることができます。精度がより必要な場合には、 **位置とサイズ** プロパティを使用するか、上記のアイテムの作成で説明したグリッドやガイドへのスナップを使用します。

複数のアイテムを同時にサイズ変更や移動する方法は、単一アイテムの場合と同様です。ただし、QGIS には、選択したアイテムのサイズを以下のさまざまなルールに基づいて自動的に変更する高度なツールが用意されています。

- 各アイテムの高さを、選択したアイテムの中で  最も大きいものにマッチさせる、あるいは  最も小さいものにマッチさせる
- 各アイテムの幅を、選択したアイテムの中で  最も広いものにマッチさせる、あるいは  最も狭いものにマッチさせる
- アイテムのサイズを  正方形に変更する：各アイテムは正方形となるように拡大されます。

同様に、複数のアイテムを均等に配置して、位置を整理する **ツール** もあります:

- アイテムの端（左右上下）
- アイテムの水平または垂直中央
- アイテムの水平または垂直方向の間隔。

アイテムのグループ化

アイテムのグループ化により、一組の複数アイテムを単一のアイテムであるかのように操作することができます。アイテム全体のサイズ変更や移動、削除、コピーが簡単に行えます。

アイテムのグループを作成するには、複数のアイテムを選択して **ビュー** メニューまたは **アクション** ツールバー、または右クリックメニューから、 **グループ** ボタンを押します。 Group という名前の行が **アイテム** パネルに追加され、他の **アイテムのパネルのオブジェクト** と同じように、ロックしたり非表示にしたりすることができます。グループ化されたアイテムに対しては、キャンバス上での選択は **個別にはできません** 。アイテムパネルを使用して直接選択し、アイテムのプロパティパネルにアクセスします。

アイテムのロック

アイテムが正しい位置に配置されたら、 **アイテム** メニューや **アクション** ツールバーの **選択アイテムをロックする** ボタンを使用するか、 **アイテム** パネルのアイテムの隣にあるボックスにチェックを入れることで、アイテムをロックすることができます。ロックされたアイテムは、キャンバスで選択 **できません** 。

ロックされたアイテムのロック解除には、 **アイテム** パネルで選んでチェックボックスのチェックを外すか、ツールバーのアイコンを使用してください。

整列と等間隔に並べる

 選択したアイテムを上へ のプルダウンメニュー内には、要素の視覚的な上下関係を上げ下げするためのツールがあります。印刷レイアウトキャンバス上の要素を選択し、使用したい機能を選択して、選択した要素を他の要素と比較して上げ下げします。この順序は **アイテム** パネルに表示されています。また、リスト内のオブジェクトのラベルをクリックしてドラッグすることでも、 **アイテム** パネル内のオブジェクトを上下させることができます。

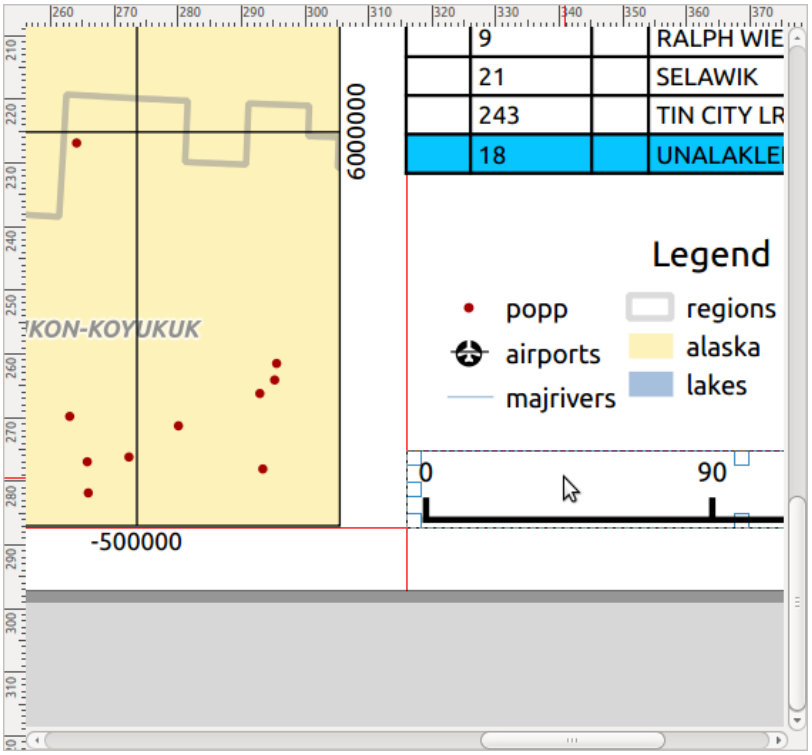


図 21.10: 印刷レイアウトの整列補助線

選択を左寄せ整列する プルダウンメニュー内には、いくつかの整列オプションがあります（ 図 21.10 参照）。整列機能を使うには、最初に要素を選択し、以下の整列アイコンのいずれかをクリックします。

- **左揃え** 、 **右揃え**
- **上揃え** 、 **下揃え**
- 水平方向の **中央揃え** 、 **垂直方向中央揃え**

選択した要素はすべて、共通のバウンディングボックスに整列します。レイアウトキャンバスでアイテムを移動させる際、境界や中央、角が揃った場合には整列補助線が現れます。

レイアウトアイテムの配置を改善するもう一つの方法は、レイアウトページ上でアイテム間の間隔を調整することです。これは、アイテムを選択し、 **左端を等間隔に並べる** ドロップダウンメニューを押して行うことができます：

- *Distribute Left Edges* or *Distribute Right Edges* of items equidistantly
- アイテムの **上端を等間隔に並べる** または **下端を等間隔に並べる**
- アイテムの **水平中央を等間隔に並べる** または **垂直中心を等間隔に並べる**
- アイテムの間に等幅の間隔を加える： **水平方向に等間隔で並べる** または **垂直方向に等間隔で並べる**

アイテムの共通プロパティ

レイアウトアイテムには、 **アイテムプロパティ** パネルの下部に、位置とサイズ、回転、フレーム、背景、アイテム ID、変数、レンダリング（ 図 21.11 参照）といった一連の共通プロパティがあります。

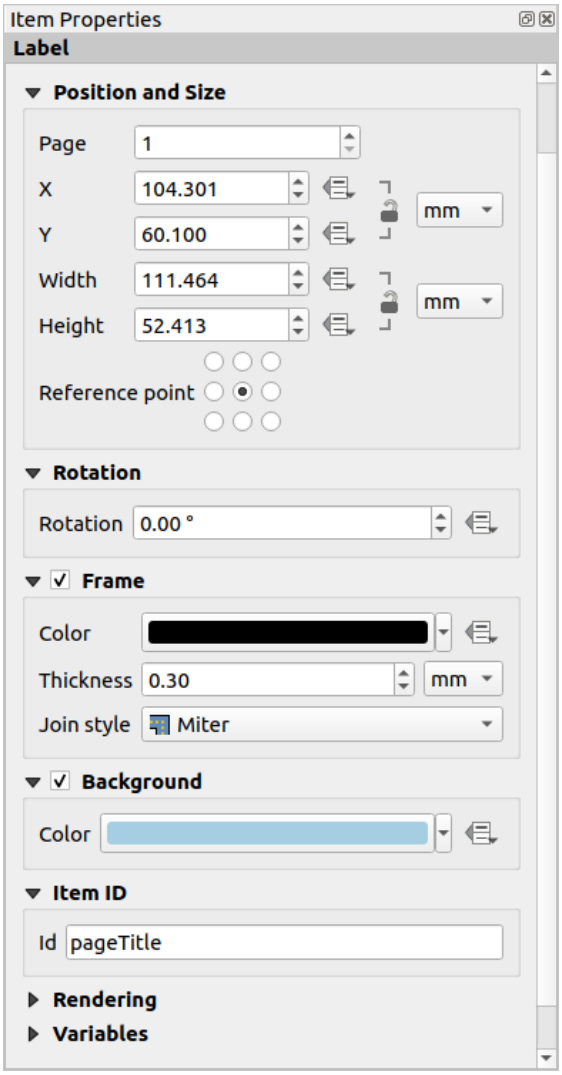


図 21.11: 共通のアイテムプロパティグループ

注釈: 大半のオプションの横にある データによって定義された上書き アイコンは、 **式** や **変数** を使用して、そのプロパティをレイヤや地物属性、ジオメトリ、あるいはその他のレイアウトアイテムプロパティと関連付けられることを意味しています。詳細な情報については、 **データによって定義された上書きの設定** を参照してください。

- **位置とサイズ** グループでは、アイテムを含むフレームのサイズと位置を定義できます（詳細は **位置とサイズ** を参照）。
- **回転** では、要素の回転を（度単位で）指定できます。
- **フレーム** は、アイテム周囲のフレームの表示・非表示を設定します。 **色** 、 **太さ** および **継ぎ目スタイル** ウィジェットを使用して、これらのプロパティを調整できます。

- **背景** メニューを使用して、背景色の設定ができます。 [色...] ボタンをクリックすると、色を選んだり、カスタム設定から選択することができるダイアログが表示されます。透過性は、不透明度フィールドの設定で調整できます。
- **アイテム** *ID* を使用すると、他の印刷レイアウトアイテムとの関係を作成できます。これは、QGIS サーバーやその他の web クライアントで使用されます。アイテム（例えば地図やラベル）に ID を設定すると、web クライアントはデータを送信して、その特定のアイテムのプロパティ（例えばラベルテキスト）を設定できます。GetProjectSettings コマンドは、レイアウトで利用可能なアイテムと ID をリストします。
- **レンダリング** モードでは、アイテムがどのように表示されるかを設定できます。具体的には **混合モード** の適用や、アイテムの不透明度の調節、 **アイテムをエクスポートから除外する** 設定ができます。

位置とサイズ

新規アイテムのプロパティ ダイアログをデータ定義の機能で拡張することで、このグループはアイテムを正確な位置に配置することができます。

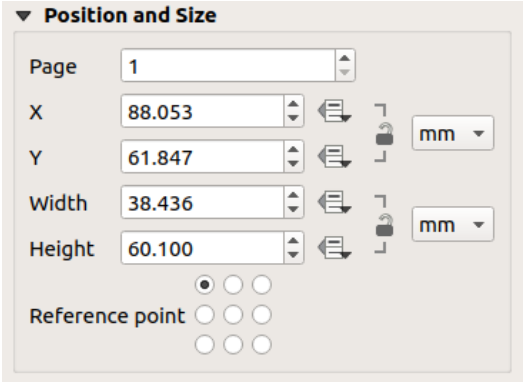


図 21.12: 位置とサイズ

- 実際にアイテムを配置するページ番号
- アイテムの基準点
- 選択したページに対するアイテムの **基準点** の *X* および *Y* 座標。これらの値の比率は、  ボタンをクリックしてロックできます。このウィジェットや  アイテムを選択/移動 ツールを使用して生じた座標値の変更は、ウィジェット、画面の両方に反映されます。
- バウンディングボックスの **幅** と **高さ** 。XY 座標と同様、幅と高さの比はロックできます。

レンダリングモード

ベクタレイヤやラスタレイヤと同様、QGIS ではレイアウトアイテムの高度なレンダリングが可能です。

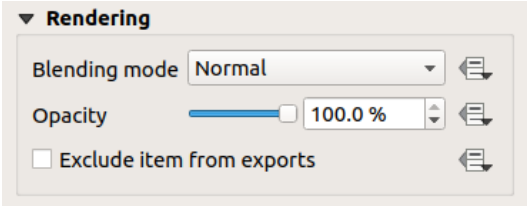


図 21.13: レンダリングモード

- **混合モード** ：このツールを使用すると、これまでグラフィックレンダリングソフトでしか実現できなかったような効果が得られます。上下のアイテムのピクセルは、設定されたモード（各効果の説明は **混合モード** を参照）に応じて混合されます。
- **不透明度**  ：このツールを使用すると、レイアウト内で下にあるアイテムを見えるようにできます。スライダーを使用して、アイテムの不透明度を必要に応じて調節します。スライダーの横にあるメニューで、可視性の割合を正確に定義することもできます。
- **アイテムをエクスポートから除外する** ：すべてのエクスポートでアイテムを非表示にすることができます。このチェックボックスを有効にすると、そのアイテムは PDF や印刷などに含まれなくなります。

変数

変数 には、レイアウトアイテムのレベルで利用可能なすべての変数がリストされています（これにはすべてのグローバル変数、プロジェクト変数、およびレイアウトの変数が含まれます）。また、地図アイテムには「地図の設定」変数もあり、地図の縮尺や範囲などの値に簡単にアクセスできます。

変数 では、アイテムレベルの変数を管理することも可能です。 ボタンをクリックして、新しいカスタム変数が追加できます。同様に、リストから任意のカスタムアイテムレベルの変数を選択して ボタンをクリックすると、それが削除されます。

変数の使用に関する詳細については、 **値を変数に格納する** のセクションを参照してください。

21.2.2 地図アイテム

地図アイテムは、あなたがマップキャンバスでデザインした地図を表示するメインフレームです。 **地図を追加** ツールを使用して、 **地図アイテムの作成手順** に従い、新しい地図アイテムを追加してください。これは、 **レイアウトアイテムの操作** と同じ方法で操作できます。

デフォルトでは、新しい地図アイテムは **マップキャンバス** の表示範囲と可視レイヤを現在の状態で表示します。これは **アイテムプロパティ** パネルを使用してカスタマイズすることができます。 **アイテムの共通プロパティ** の他に、地図アイテムには以下に示す機能があります。

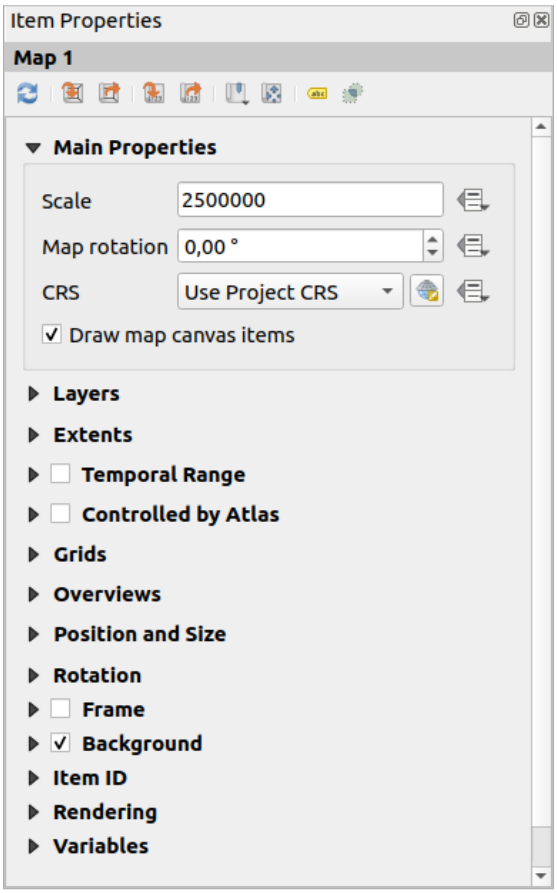


図 21.14: 地図アイテムプロパティパネル

ツールバー

地図の **アイテムプロパティ** パネルには、以下の機能を持つツールバーが埋め込まれています。

- 地図のプレビューの更新
- キャンバスの範囲に地図の範囲を合わせる
- キャンバスの地図の範囲を表示
- 地図の倍率をキャンバスの倍率に合わせる
- キャンバスを地図の倍率に合わせる
- ブックマーク ：地図アイテムの範囲を既存の空間ブックマークに一致するように設定します
- 地図範囲をインタラクティブ編集 ：地図アイテム内でパンとズームがインタラクティブにできます
- ラベルの設定 ：レイアウトのマップアイテム範囲における地物ラベルの動作（配置、可視性など）を制御します：
  - **地図の端からのマージン（余白）** はデータによる定義も可能な地図アイテムの端からの距離で、これを設定すると、距離がこの値以内ではラベルは表示されません。

- **地図の端のラベルは省略** ：これは、一部が地図アイテムの許容範囲外にあるラベルをレンダリングするかどうかを制御します。チェックを入れた場合、このようなラベルも表示されます（可視領域内に完全に見えるように配置する方法がない場合）。チェックされていない場合、部分的に見えるラベルは表示されません。
- **ラベルブロック** ：他のレイアウトアイテム（スケールバー、方位記号、地図など）を **アクティブな** 地図アイテム内の地図ラベルに対するブロッカーとしてマークすることができます。これにより、マークしたアイテムの下に地図ラベルを配置することができなくなり、ラベルの配置エンジンはこのようなラベルを別の場所に配置するよう試みるか、完全に破棄することになります。

  地図の端からのマージン（余白） が設定されている場合、地図ラベルはチェックされたレイアウトアイテムから指定距離よりも近い場所には配置されません。
- **位置未定ラベルの表示** ：位置未定のラベルを **あらかじめ定義した色** で強調表示することにより、ラベルがレイアウト地図から無くなっているかどうか（例えば、他の地図ラベルと衝突していたり、ラベルを配置する十分なスペースが無いために）を確認することができます。

- クリッピング設定 ：地図アイテムを地図帳地物やシェープアイテム、ポリゴンアイテムで切り抜くことができます。
  - **地図帳地物に切り抜く** ：レイアウト地図アイテムを現在の **地図帳地物** で自動的に切り抜くかどうかを決定できます。

    さまざまなクリッピングモードが利用可能です：

    * **レンダリング中のみクリッピング** ：ペインターベースのクリッピングを適用し、ベクタ地物で地図帳地物の外にある部分は見えなくなります。
    * **レンダリング前にクリッピング** ：地物をレンダリングする前にクリッピングを適用します。このため、一部が地図帳地物の外にある地物の境界線は、地図帳地物の境界線上に表示されます。
    * **交差地物をそのまま出力** ：現在の地図帳地物と交差する全ての地物を、そのジオメトリをクリッピングせずにレンダリングします。

    地図帳地物の中にラベルを強制移動する ことができます。地図帳地物で **すべてのレイヤを切り抜く** ことがしたくないなら、**|checkButtonOn| 選択レイヤを切り抜く** オプションを使うことができます。
  - **切り抜くアイテム** ：印刷レイアウトの *shape* アイテムや **ポリゴン** アイテムを使用して、地図アイテムの形状を変更することができます。このオプションを有効にすると、地図はコンボボックス内で選択された形状で自動的に切り抜かれます。ここでも上記のクリッピングモードが利用でき、ラベルはクリッピング形状の内側にのみ表示するよう強制させることができます。

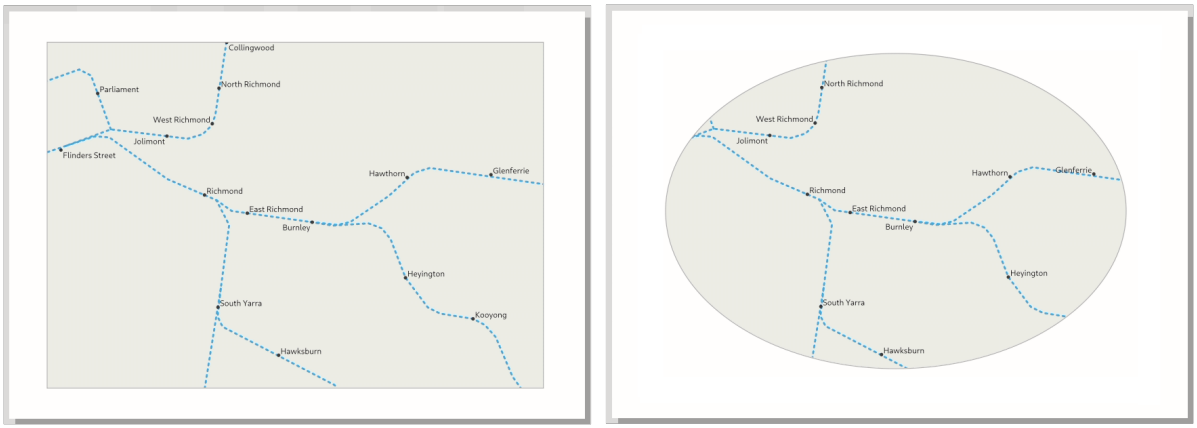


図 21.15: レイアウトの地図アイテムをシェイプで切り抜く

メインプロパティ

地図アイテムの **アイテムプロパティ** パネルの **メインプロパティ** グループ（ 図 21.14 参照）では、下記のオプションが利用可能です：

- **地図のプレビューの更新** ボタンを押すと、マップキャンバスでのビューに変更があった場合に、地図アイテムのレンダリングを更新できます。ただしほとんどの場合、地図アイテムの更新は変更によって自動的にトリガーされます。
- **縮尺** は、手動で地図アイテムの縮尺を設定します。
- **地図の回転** は、地図アイテムのコンテンツを時計回りの度単位で回転させることができます。この設定で、マップキャンバスの回転を模擬することができます。
- **座標参照系（*CRS*）** により、地図アイテムのコンテンツを任意の *CRS* で表示させることができます。デフォルトの設定は プロジェクト CRS を使う です。
- **地図キャンバスアイテムの描画** は、メインマップキャンバスに配置された **注記** を印刷レイアウトに表示させることができます。

レイヤ

デフォルトでは、地図アイテムの表示はマップキャンバスのレンダリングと同期しています。つまり、 **レイヤパネル** でレイヤの表示を切り替えたり、スタイルを変更したりすると、それが地図アイテムに自動的に適用されます。地図アイテムは他のアイテムと同様、印刷レイアウトに複数追加したい場合もあるでしょう。異なる範囲やレイヤの組み合わせ、異なるスケールなどで表示できるようにするためには、この同期を解除する必要があります。 **レイヤ** プロパティグループ（ 図 21.16 参照）では、その設定ができます。

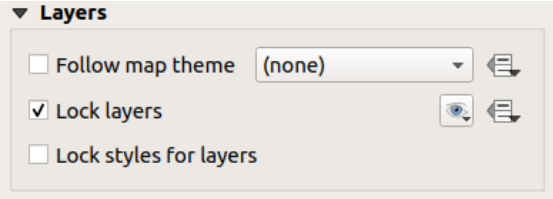


図 21.16: 地図のレイヤグループ

地図アイテムを既存の **地図テーマ** と整合させたい場合には、 **地図テーマに従う** にチェックを入れ、ドロップダウンリストから使用したいテーマを選択します。QGIS のメインウィンドウで（テーマの置き換え機能を使用して）テーマに適用された変更は、自動的に地図アイテムに影響を与えます。地図テーマが選択されているときには、 **レイヤのスタイルをロック** オプションは無効化されます。なぜなら、 **地図テーマに従う** は、レイヤのスタイル（シンボロジ、ラベル、ダイアグラム）も更新するからです。

地図アイテムで表示されているレイヤを現在のマップキャンバスの表示で固定するには、 **レイヤのロック** にチェックを入れます。このオプションが有効なときには、QGIS のメインウィンドウでのレイヤの表示に関する一切の変更は、レイアウトの地図アイテムに影響を及ぼしません。それにも関わらず、ロックしたレイヤのスタイルやラベルは QGIS のメインウィンドウに応じて更新されます。これを防ぐには、 **レイヤのスタイルをロック** を使用します。

現在のマップキャンバスを使用する代わりに、地図アイテムのレイヤを既存の地図テーマのレイヤでロックすることもできます。 地図テーマからレイヤリストを設定する ドロップダウンボタンから地図テーマを選択し、 **レイヤのロック** を有効にします。他の地図テーマを選択するか **レイヤのロック** オプションのチェックを外すまでは、地図テーマ内で表示されているレイヤの組み合わせが使用されます。 **ナビゲーション** ツールバーの ビューを更新 ボタン、あるいは上で説明した **地図のプレビューの更新** ボタンを使用して、ビューを更新する必要があるかもしれません。

なお、 **地図テーマに従う** オプションとは異なり、 **レイヤのロック** オプションが有効で地図テーマが設定されている場合には、QGIS のメインウィンドウで地図テーマが（テーマの置き換え機能を使用して）更新されたとしても、地図アイテムのレイヤは更新されません。

地図アイテムでロックされるレイヤは、オプションの横にある アイコンを使用して、 **データによって定義** することもできます。これを使用すると、ドロップダウンリストで設定された選択を上書きします。| 文字で区切ったレイヤのリストを渡す必要があります。以下の例は、地図アイテムに `layer 1` と `layer 2` レイヤのみを使用するようにロックします:

```
concat ('layer 1', '|', 'layer 2')
```

領域

地図アイテムプロパティパネルの **領域** グループには、下記の機能があります（ 図 21.17 参照）:

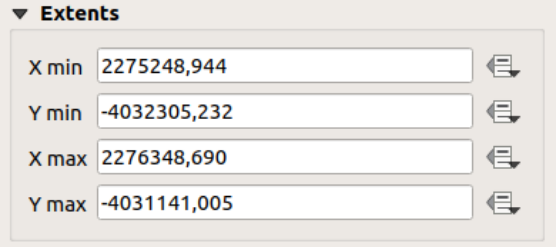


図 21.17: 地図領域グループ

領域 エリアは地図アイテムに表示されているエリアの X と Y 座標を表示します。これらの値はそれぞれ手動で置き換えることができ、マップキャンバスの表示範囲や地図アイテムのサイズを変更することができます。領域は地図アイテムパネルの上部にある以下のようなツールを使って変更することもできます：

- キャンバスの範囲に地図の範囲を合わせる
- 地図の倍率をキャンバスの倍率に合わせる

地図アイテムの範囲は アイテムのコンテンツを移動 ツールを使用しても変更することができます。地図アイテム内をクリックしてドラッグすれば、縮尺はそのままで現在のビューを修正できます。この ツールが有効なときにマウスホイールを使用して拡大・縮小を行うと、表示された地図の縮尺を変更します。 Ctrl キーを押しながら操作を行うと、小刻みな拡大・縮小ができます。

時系列範囲

地図アイテムのプロパティパネルの **時系列範囲** グループは、時間的な範囲に基づいて地図アイテム内のレイヤのレンダリングを制御するオプションを提供します。時系列プロパティが **開始** と **終了** の日付で設定された時間範囲と重なるレイヤのみが地図アイテムに表示されます。

関連するデータ定義ウィジェットは、時間範囲を動的にするのに役立ち、時間的な *atlases*、すなわち、固定された空間範囲を持ち、その内容が時間に基づいて変化する自動化されたマップを出力することを可能にします。例えば、カバレッジレイヤとして、ひとつの開始と終了のフィールドペアと、たくさんの日付範囲を表す行を持つ csv ファイルを使用し、地図アイテムプロパティで時系列範囲と地図帳による制御の両方を有効にし、地図帳のエクスポートを実行します。

地図帳による制御

地図帳による制御 グループのプロパティが利用可能となるのは、印刷レイアウトで **地図帳** が有効な場合のみです。地図アイテムを地図帳によって制御したい場合には、このオプションにチェックを入れてください。カバレッジレイヤを順に表示すると、地図アイテムの範囲は以下の設定に従い、地図帳地物にパン/ズームします。

- **地図周りの余白** ：地図アイテムの幅または高さに対するパーセンテージで表現される余白を周囲に残しながら、地図帳地物にちょうど良い縮尺でズームします。余白はすべての地物で同一とすることもできますし、 **変数の設定** により、例えば地図の縮尺に応じて設定することもできます。
- **事前定義縮尺（最適）** ：地図帳地物がもっとも適するプロジェクトの **定義済み縮尺** に合わせて、地図帳地物にズームします。

- **固定縮尺**：地図アイテムの縮尺を同じに保ったまま、地図帳地物を次々にパンしていきます。同じサイズの地図帳地物（例えばグリッド）で作業している場合や、地図帳地物間のサイズの違いを強調したい場合には最適な設定です。

グリッド

グリッドを使用すると、地図上に範囲や座標に関する情報を追加できます。この情報は地図アイテムでも、別の座標系でも可能です。 **グリッド** グループでは、地図アイテムに複数のグリッドを追加できます。

- と ボタンを使用して、グリッドを追加したり、選択したグリッドを削除したりできます。
- と ボタンを使用して、リスト内でグリッドを上下に移動させることで、地図アイテムで他のグリッドよりも前面や背後にグリッドを移動させます。

追加したグリッドをダブルクリックすると、名前を変更できます。

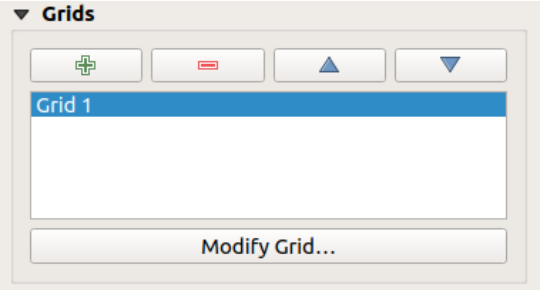


図 21.18: 地図グリッドダイアログ

グリッドを修正するには、修正したいグリッドを選択し、 **グリッドの修正...** ボタンを押して *MapGrid* **のプロパティ** パネルを開き、設定オプションにアクセスします。

グリッドの外観

MapGrid **のプロパティ** パネルでは、 **グリッドを有効にする** にチェックを入れると、地図アイテム上にグリッドを表示します。

グリッドタイプとして、以下に挙げるものを指定できます：

- **塗りつぶし**：グリッドフレームを横切る線を表示します。 **線のスタイル** で **色** や **シンボル** のセレクタウィジェットを使用して、スタイルをカスタマイズできます。
- **クロス**：グリッド線の交点の位置に線分を表示します。 **線のスタイル** と **交差幅** の設定ができます。
- **マーカー**：グリッド線の交点の位置にカスタマイズ可能なマーカーシンボルを表示します。
- **フレームと注記のみ**

グリッドタイプの他に、以下の設定の定義ができます：

- グリッドの **座標参照系（*CRS*）**。変更しない場合には、地図の CRS に従います。 *CRS* **の選択** ボタンを使用すると、グリッドに異なる CRS を設定することができます。一度設定した CRS は、座標参照

系の選択ダイアログにおいて **あらかじめ定義された** *CRS* の中にある任意のグループの見出し（例：**地理的座標系** ）を選択することで、デフォルトに戻すことができます。

- グリッドの基準として使用する **間隔** タイプの設定。利用可能な選択肢には、 地図単位 、 セグメントの幅をフィット 、 ミリメートル または センチメートル があります。
  - セグメントの幅をフィット を選択すると、マップの範囲に基づいて「ちょうど良い」グリッド間隔を動的に選択します。このオプションが選択されている場合には、間隔の 最小値 と 最大 を設定できます。
  - これ以外のオプションでは、2 つの隣り合うグリッド基準の X 方向と Y 方向の距離を設定できます。
- 地図アイテムの端からの X 方向や Y 方向への **オフセット**
- 対応している場合には、グリッドの **混合モード** （ **混合モード** 参照）

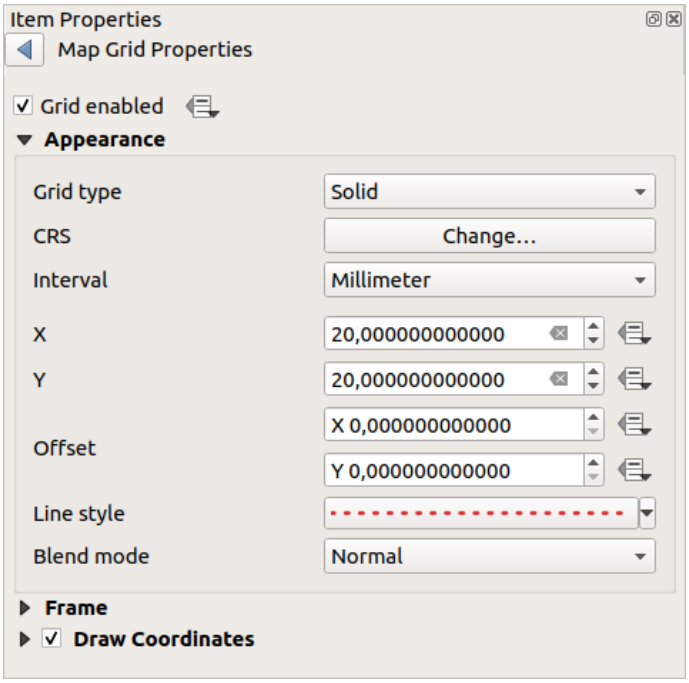


図 21.19: グリッドの外観ダイアログ

グリッドフレーム

地図の周囲のフレームのスタイル設定にはさまざまなオプションがあります。以下のオプションが利用可能です： フレームなし 、 縞 、 白黒縞（海里） 、 内側に目盛り 、 外側に目盛り 、 内側と外側に目盛り 、 境界線 、 境界線（海里）

対応しているならば、it's possible to set the **フレームサイズ** 、 **フレームマージン** 、 **フレームの線の太さ** と色、 **フレームの塗りつぶし色** の設定ができます。

目盛りセクションで 緯度/Y のみ や 経度/X のみ の値を使用すると、回転した地図や再投影されたグリッドを扱う際に、各辺に 緯度/Y と経度/X 座標が混在して表示されるのを防ぐことができます。また、グリッドフレームの各辺を表示するかしないかを選択することもできます。

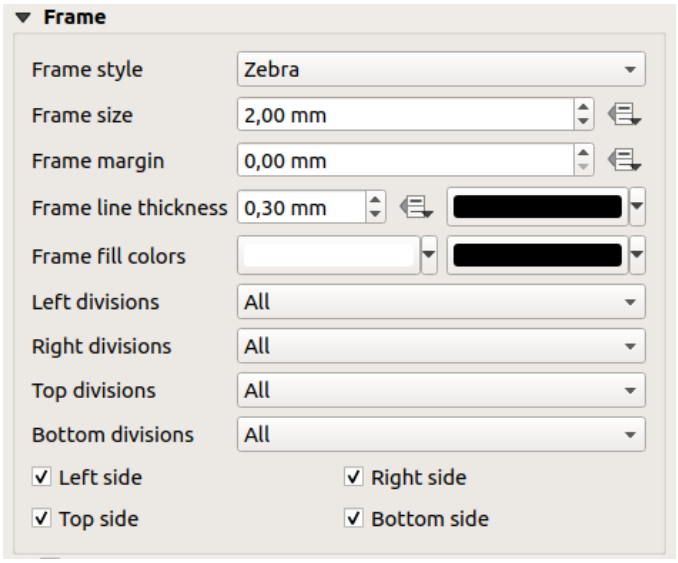


図 21.20: グリッドフレームダイアログ

座標

座標を描画 チェックボックスにチェックを入れると、地図フレームに座標を追加することができます。注釈の数値フォーマットを選択することができます。注釈の数値フォーマットを選択することができます。オプションには、数値、度、分、秒があり、単位の有無、整列の有無、そして式ダイアログを使ったカスタム形式まであります。

どの注釈を表示するかを選択することができます。選択肢には、すべて表示、緯度/Y 座標のみ、経度/X 座標のみ、または無効化（なし）です。このオプションは、地図が回転している場合に便利です。注釈は地図フレームの内側または外側に描くことができます。注釈の方向は、水平、垂直上向き、垂直下向きなどで定義できます。

最後に、注釈のフォントやフォント色、地図フレームへの距離や表示される座標の精度を定義できます。

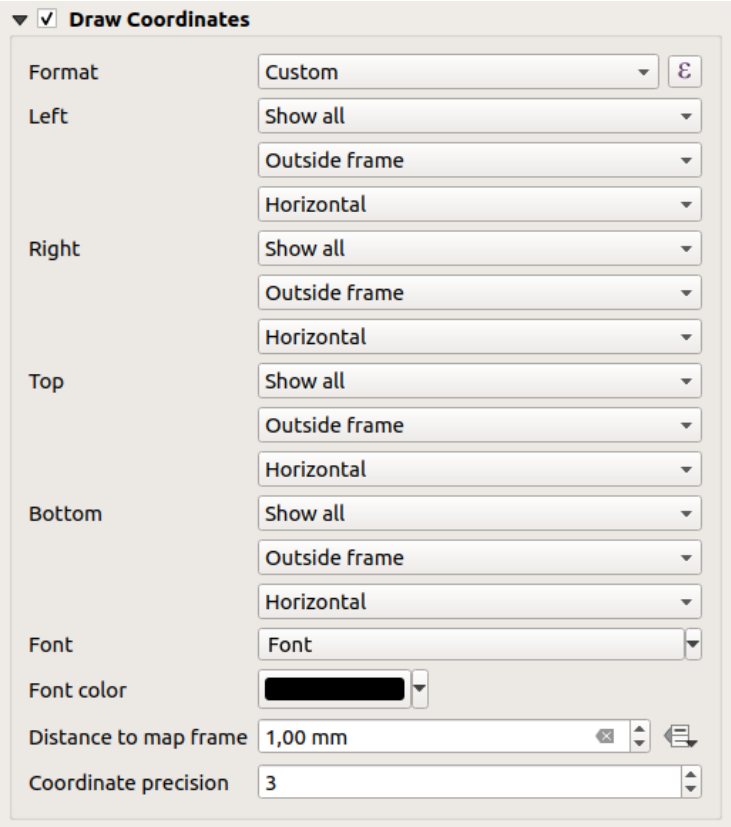


図 21.21: グリッドの座標の描画ダイアログ

全体図

時には、印刷レイアウトに複数のマップを乗せ、ある地図アイテムの調査地域を別の地図に表示したい場合もあるでしょう。これは例えば、地図を読む人が 2 つ目の地図に示されているより大きな地理的背景と関連づけて、その範囲を理解できるようにする場合です。

地図パネルの **全体図** グループでは 2 つの異なる地図の範囲間のリンクを作成することができ、以下のような機能があります。

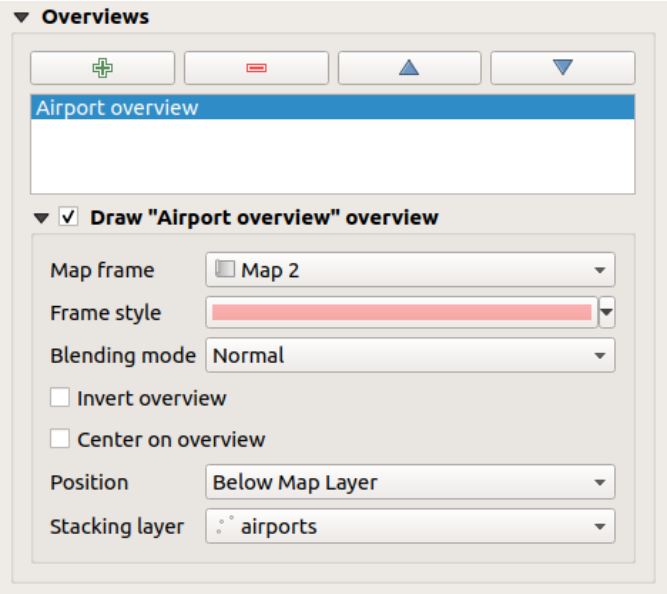


図 21.22: 地図の全体図グループ

全体図を作成するには、他の地図アイテムの範囲をその上に表示したい地図アイテムを選択し、**アイテムプロパティ** パネルの **全体図** オプションを展開します。次に ボタンを押して、全体図を追加します。

最初は、この全体図の名前は「全体図 1」となっています（ 図 21.22 参照）。ここでは、以下の操作ができます：

- ダブルクリックして名前を変更する
- と ボタンを使用して、全体図を追加・削除する
- と ボタンを使用して、リスト内で全体図を上下に移動させることで、地図アイテムで他の全体図よりも前面や背後に移動させます（同じ **スタック位置** の場合）。

次に、リストで全体図アイテムを選択し、 *"<name_overview>"* **全体図の描画** にチェックを入れて、選択した地図フレーム上への全体図の描画を有効にします。これは、以下の方法でカスタマイズできます：

- **地図フレーム** では、現在の地図アイテム上に範囲を表示する地図アイテムを選択します。
- **フレームスタイル** は、 **シンボルプロパティ** を使用して全体図フレームのレンダリングを行います。
- **混合モード** では、さまざまな透過混合モードを設定できます。
- **全体図を反転** を有効にすると、範囲の外側にマスクを作成します。参照される地図の範囲はマスクがかからずクリアに表示され、地図アイテムのその他の部分はフレームの塗りつぶし色と混合されます（塗りつぶし色を使用する場合）。
- **全体図の中心** は、地図アイテムのコンテンツをパンし、全体図フレームが地図の中心に表示されるようにします。複数の全体図がある場合、中心に使用できるのは 1 つの全体図アイテムだけです。
- **役職** は、地図アイテムのレイヤスタック内でどこに全体図が配置されるのかを正確にコントロールします。例えば、例えば、道路などの一部の地物レイヤの背後ではあるが、その他の背景レイヤよりは前面に全体図の範囲を描くことができます。利用可能なオプションは以下の通りです：
  - **マップの下**

- **レイヤの下** と **レイヤの上** ：それぞれ、あるレイヤのジオメトリの背面または前面に全体図フレームを配置します。このレイヤは **レイヤのスタック** オプションで選択できます。
- **ラベルの下** ：ラベルは常に地図アイテム内の全ての地物ジオメトリよりも前面にレンダリングされることから、全体図フレームは全ての地物ジオメトリよりも前面で、全てのラベルよりも背面に配置されます。
- **ラベルの上** ：全体図フレームを地図アイテム内の全てのジオメトリとラベルよりも前面に配置します。

21.2.3 3D 地図アイテム

3D 地図アイテムは、 *3D* **地図ビュー** を表示するために使用します。  *3D* **地図を追加** ボタンを使用して、 **アイテムの作成手順** に従い、新しい 3D 地図アイテムを追加してください。これは、 **レイアウトアイテムの操作** と同じ方法で操作できます。

デフォルトでは、新しい 3D 地図アイテムは空の状態です。3D ビューのプロパティの設定やカスタマイズが **アイテムプロパティ** パネルでできます。 **共通プロパティ** に加えて、3D 地図アイテムには以下に示す機能があります（ 図 21.23 ）

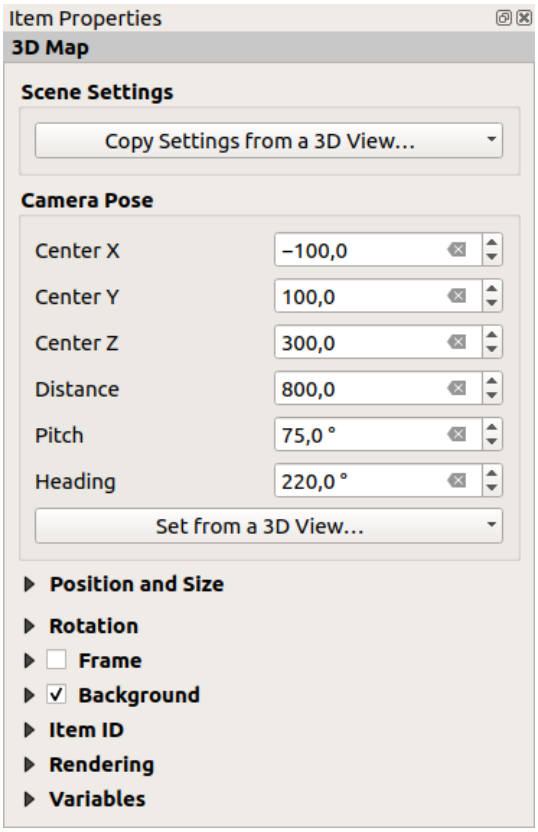


図 21.23: 3D 地図アイテムプロパティ

シーンの設定

3D **ビューから設定をコピーする**... を押すと、表示する 3D 地図ビューの選択ができます。

3D 地図ビューは、現在の構成（レイヤ、地形、光源、カメラ位置や角度など）でレンダリングされます。

カメラの位置

- *Center X* は、カメラが向いている地点の X 座標を設定します。
- *Center Y* は、カメラが向いている地点の Y 座標を設定します。
- *Center Z* は、カメラが向いている地点の Z 座標を設定します。
- **距離** は、カメラ中心からカメラが向いている地点までの距離を設定します。
- **ピッチ** は、X 軸まわりのカメラの回転（垂直方向の回転）を設定します。0 から 360（度）までの値をとります。0 ° ：真上から見た地形、90 ° ：水平（真横から）、180 ° ：真下から、270 ° ：水平で上下反転、360 ° ：真上から。
- **見出し** は、Y 軸まわりのカメラの回転（水平方向の回転、0 度から 360 度まで）を設定します。0 ° /360 ° ：北向き、90 ° ：西向き、180 ° ：南向き、270 ° ：東向き。

3D **ビューから設定**... プルダウンメニューを使用すると、3D ビューのパラメータで項目を埋めてくれます。

21.2.4 ラベルアイテム

ラベル アイテムは、地図をテキストで装飾して地図の理解を助けるためのツールです。ラベルテキストにはタイトル、作成者、データソースやその他の情報などが考えられます。 **ラベルを追加** ツールを使用して、 **アイテムの作成手順** に従いラベルを追加し、 **レイアウトアイテムの操作** と同じ方法でラベルアイテムを操作します。

デフォルトでは、ラベルアイテムにはデフォルトのテキストが入っていますが、これは **アイテムプロパティ** パネルを使用して変更できます。 **アイテムの共通プロパティ** の他に、ラベルアイテムには以下の機能があります（ 図 21.24 参照）。

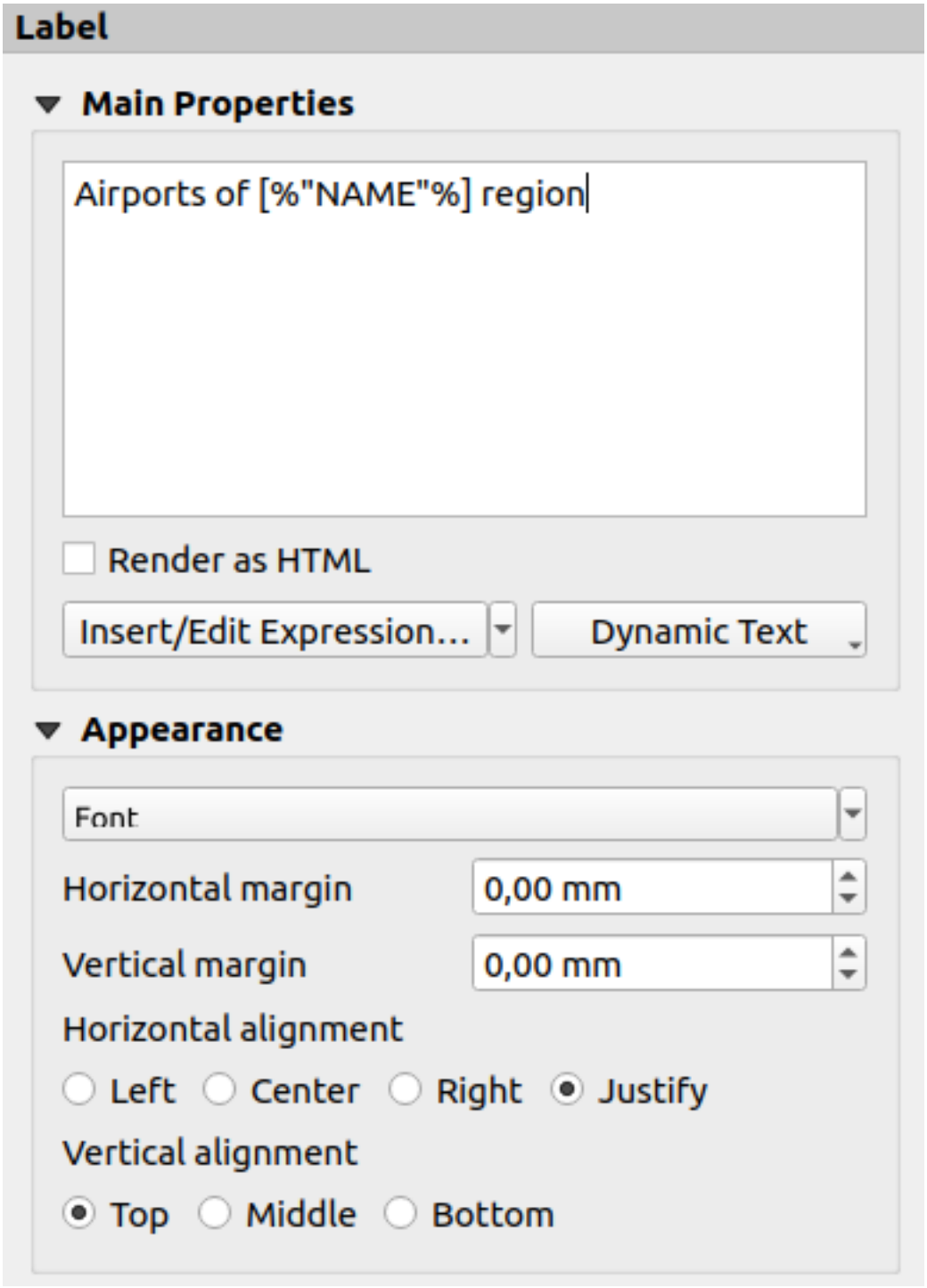


図 21.24: ラベルアイテムプロパティパネル

メインプロパティ

メインプロパティ グループはラベルのテキストを与える場所です。テキストは静的なもの、 **式** 関数や変数を使用した動的なもの、HTML で整形したものがあります。ラベルの動的な部分は解釈と評価のために [% と %] で囲み必要があります。

- ラベルで式を使用するには、**式を挿入／編集**... ボタンをクリックし、通常通りに式を記述し、ダイアログが適用されると、QGIS は自動的に前後の文字を追加します。

  ヒント: テキストボックスで何も選択せずに **式を挿入／編集**... ボタンをクリックすると、既存のテキストに新しい式が追加されます。既存の式を修正したい場合は、まず目的の部分を選択する必要があります。

  通常、地図には一般的なテキスト情報（日付、著者、タイトル、ページ番号など）が含まれているため、QGIS では対応する式や変数に直接アクセスすることができます： **動的テキスト** ボタンを押すと、それらを選択してラベルに挿入することができます。

  Tip: トップメニュー **追加** **動的テキストを追加** は、選択された定義済みの式で満たされた新しいラベルアイテムを作るのに使うことができます。

  動的ラベルを静的に変えるには: **式を挿入／編集**... ボタンの横にあるドロップダウンの矢印を押して、**静的テキストに変換** を選択してください。ラベルの内容の動的な部分が評価され、現在の値に置き換えられます。その後、必要に応じて結果テキストを手動で微調整することができます。

- ラベルは HTML コードとして解釈できます: check *HTML* **としてレンダリングする** をチェックしてください。HTML タグやスタイル、URL、ウェブページにリンクするクリック可能な画像、あるいはもっと複雑なものを挿入できるようになります。

次のコードは HTML レンダリングと式を組み合わせたもので、高度なラベリングを行い、 図 21.25 を出力します：

```
<html>
 <head>
   <style>
      /* Define some custom styles, with attribute-based size */
      name {color:red; font-size: [% ID %]px; font-family: Verdana; text-shadow: grey␣
↪1px 0 10px;}
      use {color:blue;}
   </style>
 </head>

 <body>
   <!-- Information to display -->
   <u>Feature Information</u>
   <ul style="list-style-type:disc">
     <li>Feature Id: [% ID %]</li>
```

(次のページに続く)

(前のページからの続き)

```
    <li>Airport: <name>[% NAME %]</name></li>
    <li>Main use: <use>[% USE %]</use></li>
  </ul>
  Last check: [% concat( format_date( "control_date", 'yyyy-MM-dd'), ' by <b><i>',␣
↪@user_full_name, '</i></b>' ) %]

  <!-- Insert an image -->
  <p align=center><img src="path/to/logos/qgis-logo-made-with-color.svg" alt="QGIS␣
↪icon" style="width:80px;height:50px;"</p>
 </body>
</html>
```

図 21.25: HTML スタイリングを使用したラベルの活用

外観

- **フォント** ボタンをクリックして、テキストのフォントとスタイルを定義します。**ラベルフォント** メニューでは、 **ラベルテキストの書式設定** のオプションのいくつかを使うことができます。
- 水平方向や垂直方向のマージンを mm 単位で別々に指定できます。これは、レイアウトアイテムの端からの余白です。例えばラベルアイテムを他のアイテムと位置を揃える場合など、ラベル境界の外側にラベルを配置することもできます。この場合には、マージンに負の値を使用します。
- テキスト配置の使用は、ラベルを配置するもう一つの方法です。これには、以下の設定があります：
  - **水平方向配置** については、 **左** 、 **中央部** 、 **右** 、 **正当化する**
  - **垂直方向配置** については、 **上部** 、 **中央部** 、 *Bottom*

ラベルアイテムの式の探究

以下は、ラベルアイテムに興味深い情報入力するために使用できる式のいくつかの例です。コード、あるいは少なくとも計算式の部分は、 **メインプロパティ** フレームで [% と %] で囲む必要があります。

- 現在の地図帳地物の "field1" の値を使ったタイトルを表示する：

```
'This is the map for ' || "field1"
```

メインプロパティ セクションに書く場合には、以下のようにします：

```
This is the map for [% "field1" %]
```

- 現在の地図帳地物に関するページ番号を追加する（例 `Page 1/10`）：

```
concat( 'Page ', @atlas_featurenumber, '/', @atlas_totalfeatures )
```

- 共通の属性に基づいて、現在の地図帳領域地物に対する空港の名前を返す：

```
aggregate( layer := 'airports',
           aggregate := 'concatenate',
           expression := "NAME",
           filter := fk_regionId = attribute( @atlas_feature, 'ID' ),
           concatenator := ', '
         )
```

属性リレーション が設定されている場合には、次のようにもできます：

```
relation_aggregate( relation := 'airports_in_region_relation',
                    aggregate := 'concatenate',
                    expression := "NAME",
                    concatenator := ', '
                  )
```

- 空間関係に基づいて、現在の地図帳領域地物に含まれる空港の名前を返す：

```
aggregate( layer := 'airports',
           aggregate := 'concatenate',
           expression := "NAME",
           filter := contains( geometry( @parent ), $geometry ),
           concatenator := ', '
         )
```

または:

```
array_to_string( array:= overlay_contains( layer := 'airports',
                                           expression := "NAME" ),
                 delimiter:= ', '
               )
```

- `Map 1` アイテムの範囲の X 座標の最小値を返す：

```
x_min( map_get( item_variables( 'Map 1' ), 'map_extent' ) )
```

- 現在のレイアウトの `Map 1` アイテムのレイヤの名前を取得し、1 行に 1 つのレイヤ名となるよう整形する：

```
array_to_string(
 array_foreach(
  map_get( item_variables( 'Map 1' ), 'map_layers' ), -- retrieve the layers list
  layer_property( @element, 'name' ) -- retrieve each layer name
 ),
 '\n' -- converts the list to string separated by breaklines
)
```

- レイヤのリストとライセンス文字列（使用権）をレイアウト Map 1 アイテムに表示します。最初にレイヤの アクセス メタデータ プロパティを入力する必要があります。

```
array_to_string( map_credits( 'Map 1', true ) )
```

21.2.5 凡例アイテム

凡例 アイテムは、地図上で使用されているシンボルの意味を説明するボックス、あるいはテーブルです。このため、凡例は地図アイテムに関連付けられます。凡例アイテムは **アイテムの作成手順** に従って **凡例を追加** ツールを使用して追加でき、 **レイアウトアイテムの操作** と同じ方法で操作できます。

デフォルトでは、凡例アイテムは全ての利用可能なレイヤを表示しますが、これは **アイテムプロパティ** パネルを使用して改良できます。 **アイテムの共通プロパティ** の他に、凡例アイテムには以下の機能があります（ 図 21.26 参照）。

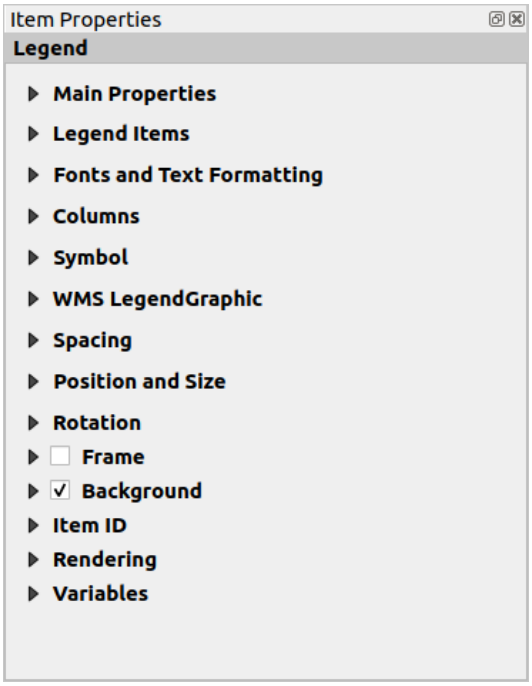


図 21.26: 凡例アイテムプロパティパネル

メインプロパティ

凡例の **アイテムプロパティ** パネルの **メインプロパティ** グループには、下記の機能があります（ 図 21.27 参照）：

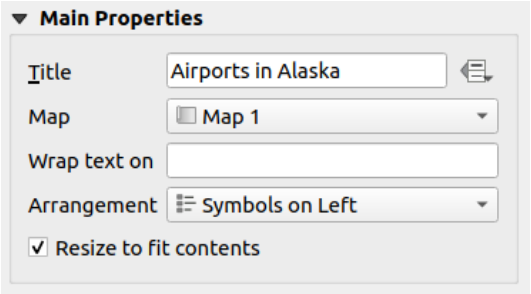


図 21.27: 凡例メインプロパティグループ

メインプロパティでは、以下の設定ができます。

- 凡例の **タイトル** の変更。 **データによって定義された上書き** の設定を使用すれば、タイトルを動的に変更することもでき、例えば地図帳の作成時に便利です。
- 現在の凡例がどの **地図** アイテムを参照しているかの選択。デフォルトでは、描画した凡例アイテムの下にある地図が選ばれます。もし下に地図アイテムが無ければ、 **参照マップ** にフォールバックします。

  注釈: リンクしている地図アイテムの **変数**（@map_id, @map_scale, @map_extent...）は、凡例のデータ定義のプロパティからもアクセス可能です。

- 指定した文字による凡例テキストの折り返し。凡例テキストにその文字が現れるたびに、改行文字で置き換えられます。
- 凡例内でのシンボルとテキストの配置の設定。 **配置** は、 **左側のシンボル** または **右側のシンボル** とすることができます。デフォルト値は使用しているロケール（右から左へと書く言語か、そうでないか）によります。
- **大きさを内容に合わせる** を使用すると、コンテンツに合わせて凡例を自動的にサイズ変更するかどうかを制御できます。チェックしない場合には凡例のサイズ変更は行われず、ユーザーが設定した大きさに固定されます。サイズが合わないコンテンツはすべて切り取られます。

凡例アイテム

凡例の **アイテムプロパティ** パネルの **凡例アイテム** グループには、下記の機能があります（ 図 21.28 参照）：

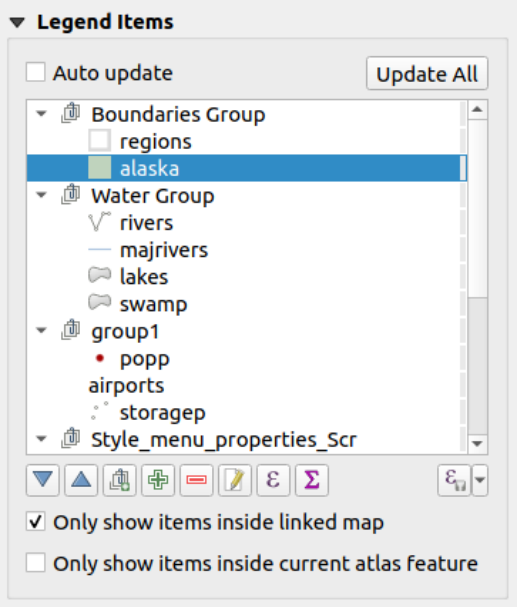


図 21.28: 凡例アイテムグループ

- **自動更新** にチェックが入っていると、凡例は自動的に更新されます。When **自動更新** がチェックされていない時には、凡例アイテムをより細かくコントロールすることができます。凡例アイテムのリスト下にある全てのアイコンが有効です。
- 凡例アイテムウィンドウにはすべての凡例アイテムが一覧表示されており、アイテムの順番の変更やレイヤのグループ化、リスト内のアイテムの削除と復元、レイヤ名とシンボロジの編集、フィルタの追加などができます。
  - や ボタンを使用するか、ドラッグ＆ドロップでアイテムの順序を変更できます。 WMS 凡例グラフィックは順序を変更できません。
  - ボタンを使用して凡例のグループを追加します。
  - ボタンを使用してレイヤを追加し、 ボタンを使用してグループ、レイヤまたはシンボルクラスを削除します。
  - ボタンを使用して、レイヤ名やグループ名、タイトルを編集できます。最初に凡例アイテムを選択する必要があります。アイテムをダブルクリックしても、テキストボックスが開き、名前の変更ができます。
  - ボタンは、選択したレイヤの各シンボルラベルをカスタマイズするために式を使用します（**データ定義による凡例ラベル** 参照）
  - ボタンは、ベクタレイヤの各クラスの地物数を凡例に追加します。
  - 式による凡例フィルタ は、レイヤのどの凡例アイテムを表示するかをフィルタリングするのに役立ちます。例えば、異なる凡例アイテムを持つレイヤ（ルールに基づいた、あるいはカテゴリ値によるシンボロジなど）を使って、条件式を満たす地物が無い場合にはそのスタイルを凡例ツリーから除外するためのブール式を指定することができます。ただし、そのような地物はレイアウトの地図アイテムでは残っており、表示されることに注意してください。

凡例アイテムのデフォルトの動作は、 **レイヤ** パネルツリーを模倣し、同じグループ、レイヤ、シン

ボロジのクラスを表示しますが、アイテムを右クリックするとレイヤ名を隠したり、グループやサブグループとして表示するオプションがあります。レイヤに何らかの変更を加えた場合は、凡例エントリのコンテキストメニューから **デフォルトにリセット** を選択することで、変更を元に戻すことができます。

QGIS のメインウィンドウでシンボロジを変更したあと、 **全て更新** ボタンをクリックすると、印刷レイアウトの凡例要素に変更が反映されます。

- **リンク先地図の内側にあるアイテムだけ表示する** にチェックを入れると、リンクした地図で表示されている凡例アイテムのみが凡例にリストされます。このツールは、 **自動更新** が有効な場合にも利用可能です。
- ポリゴン地物を使用して地図帳を作成する際に、現在の地図帳地物の外にある地物の凡例アイテムをフィルタして除外できます。これを行うには、 **現在の地図帳地物の内部のアイテムのみを表示する** オプションにチェックを入れてください。

データ定義による凡例ラベル

により、与えられたレイヤの各シンボルラベルに **式** を追加できます。新しい変数（ `@symbol_label` 、 `@symbol_id` と `@symbol_count` ）は、凡例エントリを操作する上で役立ちます。

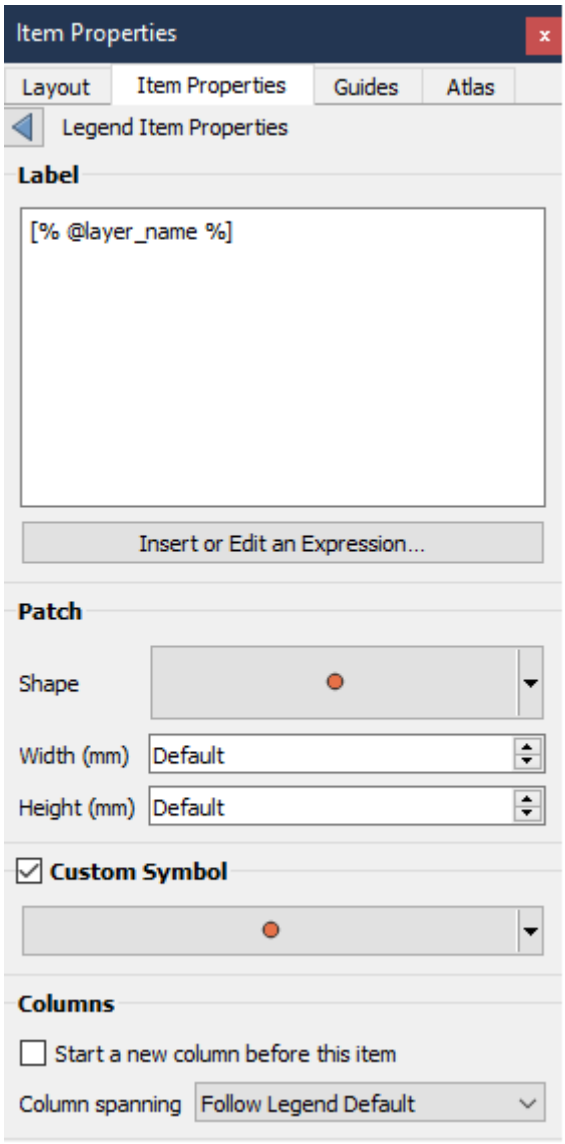
例えば、 `regions` レイヤがあり、 `type` フィールドで分類されているとしましょう。この凡例の各クラスに、 `Borough (3) - 850ha` のように地物の数と面積を追加できます：

1. 凡例ツリー内で設定したいレイヤのエントリを選択します。
2. ボタンを押し、 **式文字列ビルダ** ダイアログを開きます。
3. 以下の式を入力します（ **シンボルラベルは編集されていないことを仮定しています** ）：

```
format( '%1 (%2) - %3ha',
        @symbol_label,
        @symbol_count,
        round( aggregate(@layer, 'sum', $area, filter:= "type"=@symbol_label)/
↪10000 )
      )
```

4. *OK* ボタンを押します。

凡例をカスタマイズする



凡例アイテムは **凡例アイテムプロパティ** で個別にカスタマイズすることもできます。しかし、これらのカスタマイズは **自動更新** が無効になっている場合のみ可能です。

アイテムをダブルクリックするか、アイテムのプロパティを編集 を押すと、さらにカスタマイズすることができます。

ラベル

すべてのアイテムタイプについて、**式の挿入・編集** を使って式を入力または挿入することでラベルのテキストを変更することができます。式は [%式 %] 記法を使ってアイテムのラベルの任意の場所に直接追加することもできます。

カラム

凡例プロパティは、列分割を特定のアイテムの後またはレイヤのすべてのシンボルに強制的に発生させることによって、列分割の動作を制御することもできます。このウィジェットではレイヤを基にレイヤとそ

の子の自動分割を許可またはブロックすることもできます。

パッチ

シンボルを持つアイテムの場合、凡例アイテムプロパティでシンボルが占有できる最大の高さと幅を指定できます。

ベクタシンボルの場合、シンボルにカスタムシェイプを指定することができます。シェイプは通常、単純な平面でジオメトリを表現する式で定義されますが、これらのシンボルはスタイルマネージャに保存して後でインポートすることもできます。各ジオメトリタイプのデフォルトシンボルもスタイルマネージャで制御できます。

カスタムシンボル

カスタムシンボルはベクタシンボルにも指定できます。これは特定のシンボルのレンダリングを微調整したり、凡例でそのシンボルを強調したり、真のシンボルのプレビューから独立したシンボルを持つのに便利です。このカスタムシンボルは凡例シンボルを上書きしますが、指定されたシンボル **パッチ** を考慮します。

フォント

凡例の **アイテムプロパティ** パネルの **フォント** グループには、下記の機能があります：

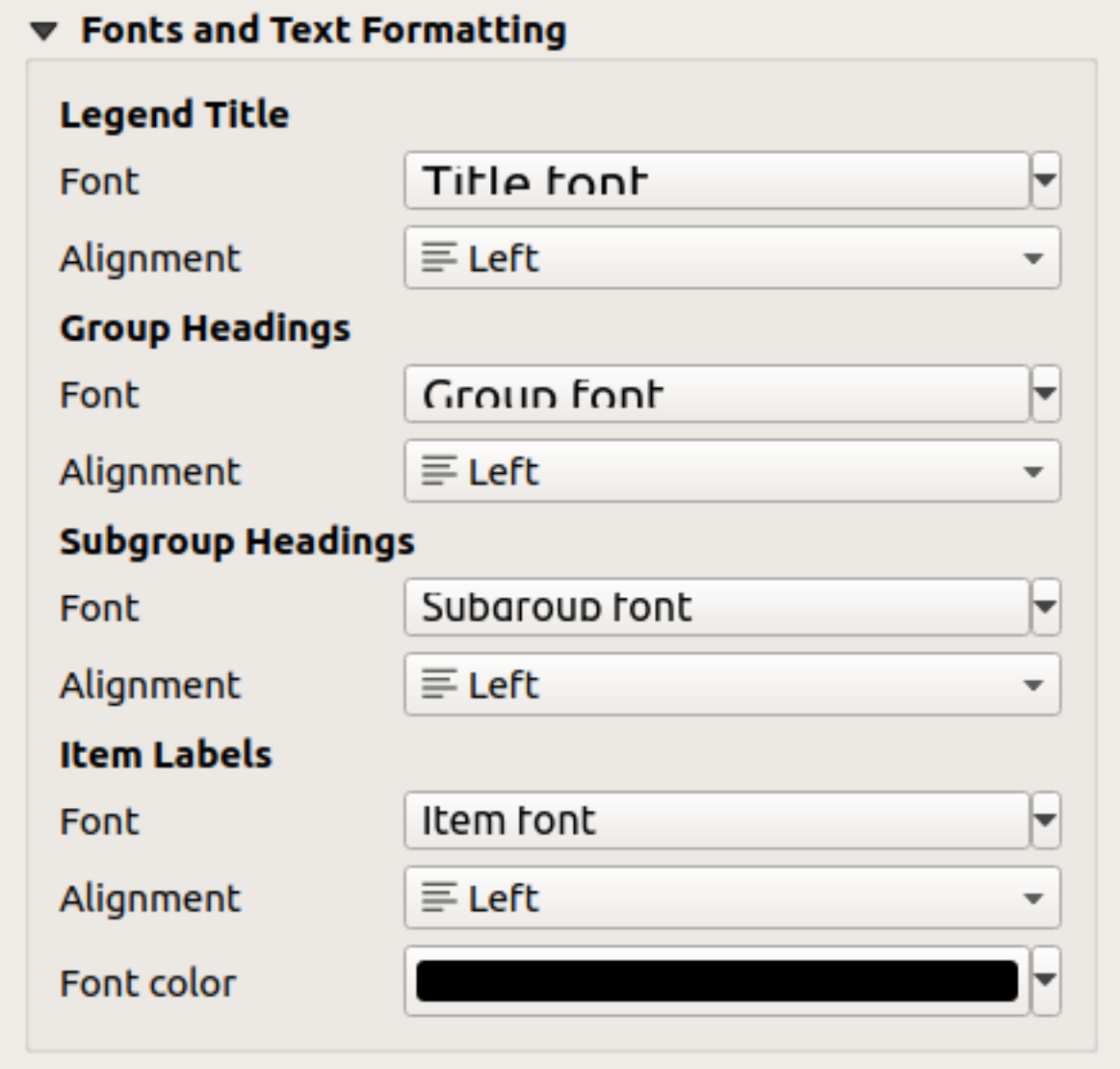


図 21.29: 凡例フォントプロパティ

- **フォントセレクタ** ウィジェットを使用して、凡例アイテム内の凡例タイトル、グループ見出し、サブグループ見出しとアイテム（地物）のフォントを変更できます。
- これらの各レベルで、テキストの **整列** の設定ができます。設定値には、**左**（左から右に文字を書くロケールでのデフォルト）、**中央**、または **右**（右から左に文字を書くロケールでのデフォルト）があります。
- **色セレクタ** ウィジェットを使用して、ラベルの **色** を設定できます。選択した色は、凡例内の全てのフォントアイテムに適用されます。

カラム

凡例の **アイテムプロパティ** パネルの **カラム** グループでは、凡例アイテムを複数の列にわたって並べる設定ができます：

- **カウント（*Count*）** 1,00 フィールドに列の数を設定します。この値は、例えば地図帳地物や凡例のコンテンツ、フレームのサイズ等に応じて動的に変更させることもできます。
- **等幅** は、凡例の列幅がどのように調整されるかを設定します。
- **レイヤを分割** オプションは、カテゴリ値や連続値によるレイヤの凡例を列にまたがって分割することができます。

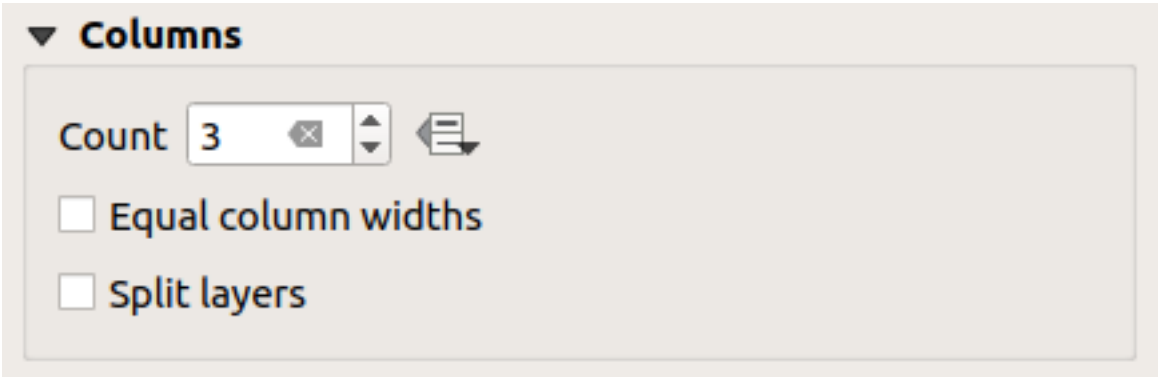


図 21.30: 凡例のカラムの設定

シンボル

凡例の **アイテムプロパティ** パネルの **シンボル** グループでは、凡例ラベルの横に表示されるシンボルのサイズを設定します。以下の設定ができます：

- **シンボル幅** と **シンボル高さ** の設定
- マーカーの **最小シンボルサイズ** と **最大シンボルサイズ** を設定します: `0.00mm` は値が設定されていないことを意味します。
- **ラスタシンボルのストローク（輪郭）を描画** ：これは、シンボルにラスタレイヤのバンドカラーを表す輪郭線を追加します。 **ストローク色** と **太さ** を設定できます。

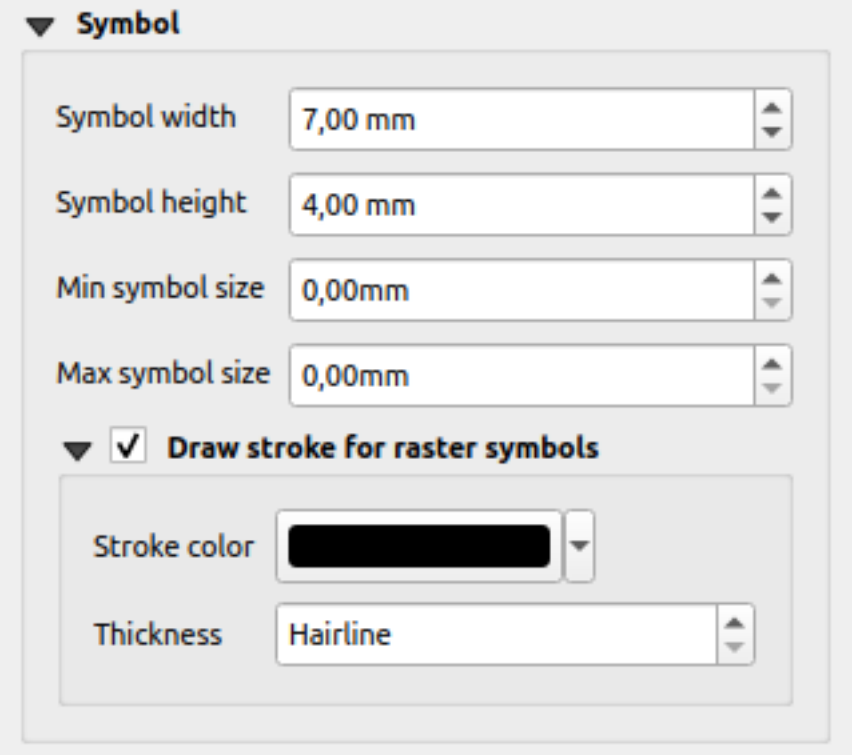


図 21.31: 凡例のシンボル設定

WMS LegendGraphic

凡例の **アイテムプロパティ** パネルの *WMS LegendGraphic* セクションは以下の機能を提供します（ 図 21.32 を参照）：

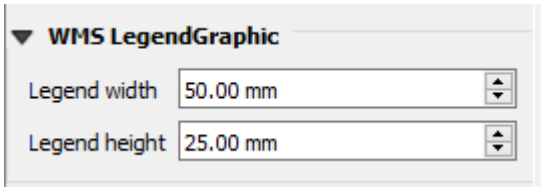
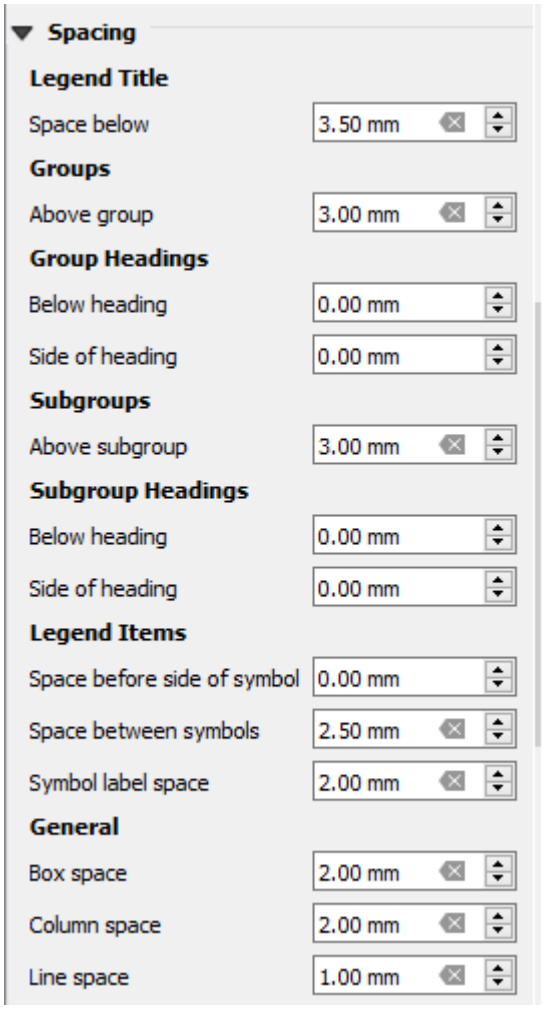


図 21.32: WMS LegendGraphic

WMS レイヤを追加して凡例アイテムを挿入すると、WMS の凡例を提供するよう WMS サーバーにリクエストが送信されます。この凡例は、WMS サーバーが GetLegendGraphic ケーパビリティを提供している場合にのみ表示されます。WMS 凡例のコンテンツはラスタ画像として提供されます。

WMSLegendGraphic は、WMS 凡例のラスタ画像の **凡例の幅** と **凡例の高さ** を調整するために使用します。

間隔



間隔 セクションでは凡例内の間隔をカスタマイズすることができます。間隔は凡例内のアイテムのグループ化とそれらの関係を示すのに大いに役立ちます。

タイトル、グループ、サブグループ、シンボル、ラベル、ボックス、カラム、行の前後の **間隔** はこのダイアログでカスタマイズできます。

21.2.6 スケールバーアイテム

スケールバーは、地図アイテム上の地物のサイズや地物間の距離を視覚的に表示するものです。スケールバーアイテムには、地図アイテムが必要です。 **スケールバーを追加** ツールを使用して、 **アイテムの作成手順** に従い、新しいスケールバーアイテムを追加してください。これは作成後、 **レイアウトアイテムの操作** と同じ方法で操作できます。

デフォルトでは、新しいスケールバーアイテムは、その下にある地図アイテムのスケールを表示します。地図アイテムが下に無い場合には、 **参照マップ** を使用します。これは **アイテムプロパティ** パネルでカスタマイズすることができます。 **アイテムの共通プロパティ** の他に、スケールバーアイテムには以下の機能があります（ 図 21.33 参照）。

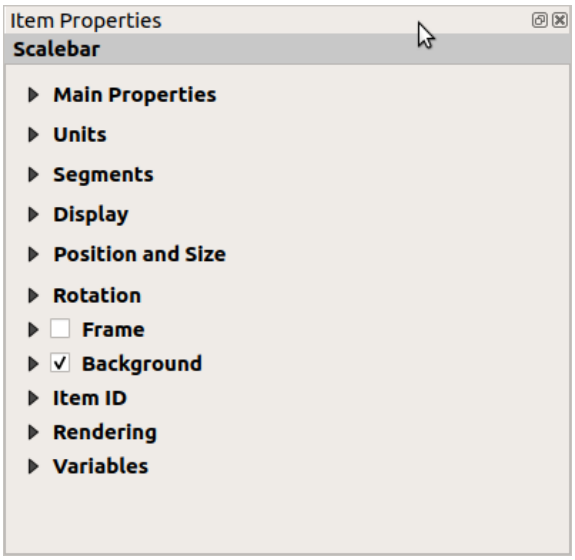


図 21.33: スケールバーアイテムプロパティパネル

メインプロパティ

スケールバーの **アイテムプロパティ** パネルの **メインプロパティ** グループには、以下の機能があります（ 図 21.34 参照）：

図 21.34: スケールバーのメインプロパティグループ

1. まずは、スケールバーを関連付ける地図を選択します。
2. 次に、スケールバーのスタイルを選択します。利用可能なスタイルは、
   - **シングルボックス** と **ダブルボックス** スタイルは、交互に色が並んだ 1 行または 2 行のボックスです。
   - **中央チック** 、 **上向きチック** 、 **下向きチック**
   - **ステップ線** スタイルは、スケールバー表現として階段状の線を表示します。
   - **窪み箱** スタイルは、交互に色分けされたシングルボックスのセグメントを描画し、1 つおきのセグメントには水平線が引かれます。
   - **数値** は、縮尺の比を表示します（例 `1:50000` ）
3. プロパティを適切に設定します。

単位

スケールバーの **アイテムプロパティ** パネルの **単位** グループには、 表示する単位とテキストフォーマットの設定があります（ 図 21.35 参照）。

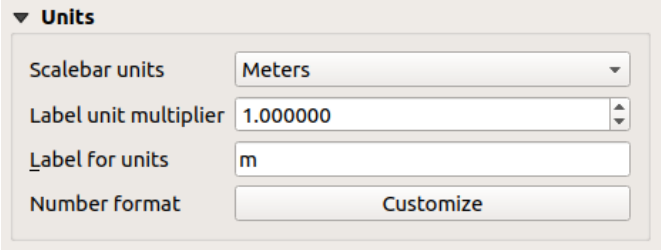


図 21.35: スケールバーの単位グループ

- **スケールバーの単位** として使用したい単位を選択します。 **地図上の単位** （デフォルト）、 **メートル** 、 **フィート** 、 **マイル** 、 **海里** や、その派生単位など、多数の選択肢があります。単位の変換は自動的に行われます。
- **ラベル単位の乗数** は、ラベルの 1 単位あたりのスケールバー単位の数を指定します。例えば、スケールバーの単位が "メートル" に設定されている場合、乗数を 1000 とすると、スケールバーのラベルを "キロメートル" の数字で表示することになります。
- **単位のラベル** フィールドは、 m や km のように、スケールバーの単位を表すテキストを定義します。これは上記の乗数を反映して合うように定義してください。
- **数値フォーマット** の横にある **カスタマイズ** ボタンを押すと、スケールバーの数字に関する全ての書式プロパティを制御することができます。これには、数字の 3 桁区切りや小数点以下桁数、有効数字表記などが含まれます（詳細については **数値フォーマット** 参照）。現在の QGIS のロケールとは異なるロケールの閲覧者向けに地図を作成する場合や、ロケールのデフォルトスタイルから変更したい場合（例えばロケールのデフォルトでは 3 桁区切りを表示しないが、スケールバーでは 3 桁区切り記号を追加したい場合）に便利です。

セグメント

スケールバーの **アイテムプロパティ** パネルの **セグメント** グループには、セグメントの数やサイズ、小分割の設定をするための機能があります（ 図 21.36 参照）：

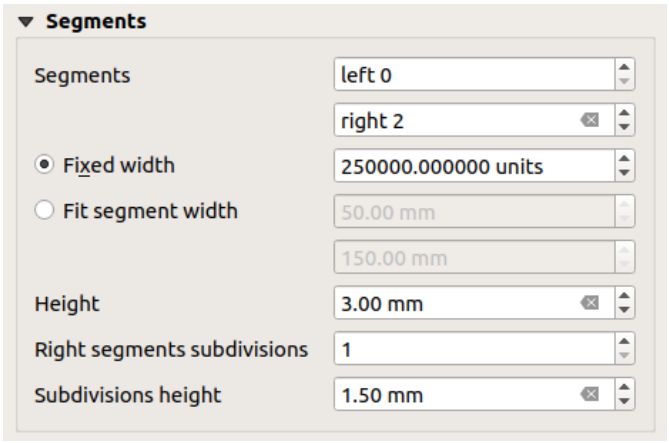


図 21.36: スケールバーのセグメントグループ

- スケールバーの `0` の左右に描画される **セグメント** の数を定義できます。
  - **左** 側にある単一セグメントの小分割の数
  - **右** 側にあるセグメントの数
- セグメントの幅またはスケールバーの全長の範囲を設定できます：
  - スケールバーの単位でセグメントの長さを設定します (**固定幅**)
  - または **セグメント幅をフィット** オプションを使って、スケールバーのサイズを mm 単位で制限します。後者の場合、地図の縮尺が変わるたびに、設定された上下の範囲に収まるようにスケールバーのサイズが変更されます（ラベルも更新されます)。
- **高さ** は、バーの高さを定義します。
- **右側の小分割** は、スケールバーの右側のセグメントにある区切りの数を定義します（ **下向きチック** 、 **中央チック** 、 **上向きチック** のスケールバースタイルで利用できます)。
- **小分割の高さ** は、小分割セグメントの高さを定義します。

表示名

スケールバーの **表示名** グループの **アイテムプロパティ** パネルには、下記の機能があります：

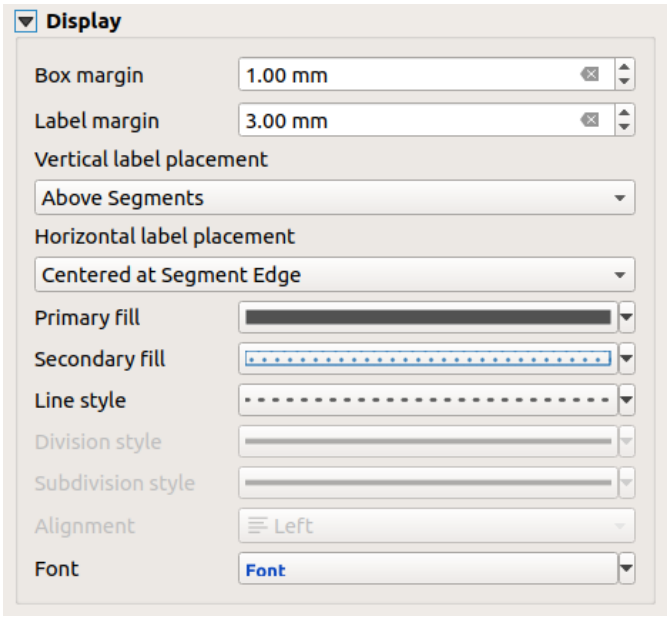


図 21.37: スケールバーの表示名グループ

フレーム内でスケールバーをどのように表示するか定義できます。

- **ボックスのマージン** ：テキストとフレーム境界との間隔です。
- **ラベルのマージン** ：テキストとスケールバーとの間隔です。
- **垂直配置** ：スケールバーセグメントの上または下にラベルを配置できます。
- **水平配置** ：スケールバーセグメントの端または中央にラベルの中心が来るようにします。
- スケールバー図形の **第一塗りつぶし** と **第二塗りつぶし** は、 **塗りつぶしシンボルプロパティ** を使用します（色、不透明度、描画エフェクト等）。 **シングルボックス** 、 **ダブルボックス** 、 **窪み箱** スタイルで使用できます。
- スケールバー図形の **線のスタイル** は、 **ラインシンボルプロパティ** （色、ストロークスタイル、継ぎ目スタイル、両端スタイル、パターン、描画エフェクト等）を使用します。 **数値** スタイル以外の全てで使用できます。
- **分割スタイル** と **小分割スタイル** は、 **上向きチック** 、 **中央チック** 、 **下向きチック** のスケールバースタイルにおける分割セグメントや小分割セグメントのスタイル設定に **ラインシンボルプロパティ** （色、ストロークスタイル、継ぎ目スタイル、両端スタイル、パターン、描画エフェクト等）を使用します。
- **整列** は、フレームの左、中央、または右にテキストを配置します（ **数値** スケールバースタイルのみ）
- **フォント** は、スケールバーのラベルの **プロパティ** （大きさ、フォント、色、間隔、影、背景など）を設定します。

スケールバーの表示名プロパティの多くはデータ定義可能なプロパティのシンボルに依拠していることから、スケールバーをデータ定義で表示させることが可能です。

例 ：スケールバーのラベルの太字プロパティに以下のコードを適用すると、数値が 500 の倍数のときに太字で表示します：

```
-- returns True (or 1) if the value displayed on the bar
-- is a multiple of 500

@scale_value % 500 = 0
```

21.2.7 テーブルアイテム

テーブルアイテムを使用して、地図を装飾したり、説明を加えたりすることができます：

- **属性テーブル** ：あらかじめ定義したルールに基づいて、レイヤの属性テーブルのサブセットを表示します。
- **固定テーブル** ：レイヤとは独立した情報を持つことができる、手入力するテキストテーブルを挿入します。

属性テーブルアイテム

プロジェクトの任意のレイヤは、印刷レイアウト内で属性を表示させることができます。  **属性テーブルを追加** ツールを使用して、 **アイテムの作成手順** に従い、新しいテーブルアイテムを追加してください。これは、 **レイアウトアイテムの操作** と同じ方法で操作できます。

デフォルトでは、新しい属性テーブルアイテムは（アルファベット順で）最初のレイヤの全属性の 1 行目を読み込みます。ただしこれは、属性テーブルの **アイテムプロパティ** パネルを使用して、テーブルのカスタマイズができます。 **アイテムの共通プロパティ** の他に、属性テーブルアイテムには以下の機能があります（ 図 21.38 参照）。

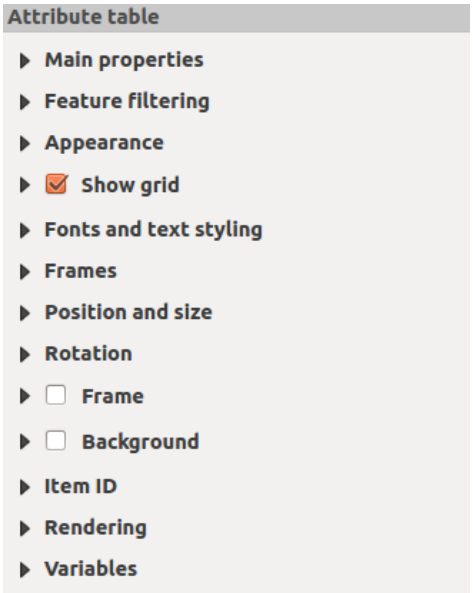


図 21.38: 属性テーブルアイテムプロパティパネル

メインプロパティ

属性テーブルの **メインプロパティ** グループには、以下の機能があります（ 図 21.39 参照）：

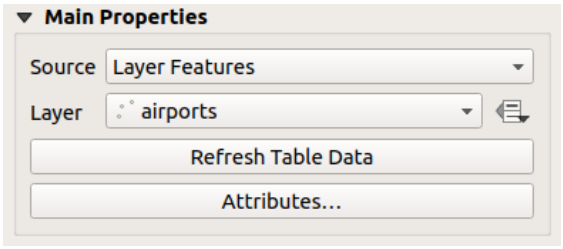


図 21.39: 属性テーブルのメインプロパティグループ

- **ソース** として、デフォルトでは **レイヤ地物** のみを選択でき、プロジェクトで読み込んだベクタレイヤの中から **レイヤ** を選択します。

  レイヤリストの近くにある、 データによって定義された上書き ボタンを使うと、テーブルの入力に使用するレイヤを動的に変更できます。例えば、地図帳のページ毎に異なるレイヤ属性で属性テーブルを埋めることができます。使用されるテーブル構造（ 図 21.42 ）には、 **レイヤ** ドロップダウンリストに表示されているレイヤのテーブル構造が用いられ、これは変更できないことに注意してください。つまり、データ定義テーブルをレイヤに設定し、別のフィールドをデータ定義で表示させようとすると、テーブル内には空のカラムができてしまいます。

  地図帳 パネル（ **地図帳の作成** 参照）で **地図帳を作成する** オプションを有効にする場合には、**ソース** には他に 2 つの選択肢があります。

  - **地図帳 個の地物** （ 図 21.40 参照）：レイヤを選択するオプションは表示されず、地図帳カバレッジレイヤの現在の地物の属性をテーブルアイテムの行に表示するのみです。
  - **リレーションの子要素** （ 図 21.41 参照）：関係名を指定するオプションが現れます。この機能は、地図帳カバレッジレイヤを親とした **リレーション** を定義している場合にのみ使用することができ、テーブルは地図帳カバレッジレイヤの現在の地物の子要素の行が表示されます。

- 実際のテーブルコンテンツに変更があった場合には、 **テーブルデータのリフレッシュ** ボタンを使用してテーブルの更新ができます。

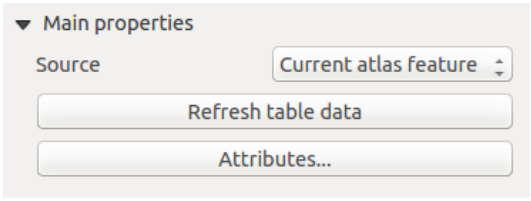


図 21.40: 「地図帳 個の地物」の属性テーブルメインプロパティ

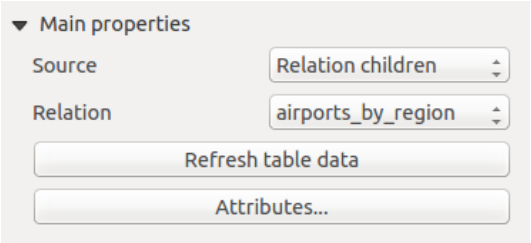


図 21.41: 「リレーションの子要素」の属性テーブルメインプロパティ

- **属性**... ボタンを押すと **属性の選択** ダイアログ（図 21.42 参照）が開き、テーブルで表示されるコンテンツの変更ができます。ウィンドウの上半分には表示する属性のリストが表示され、下半分ではデータの並び替えができます。

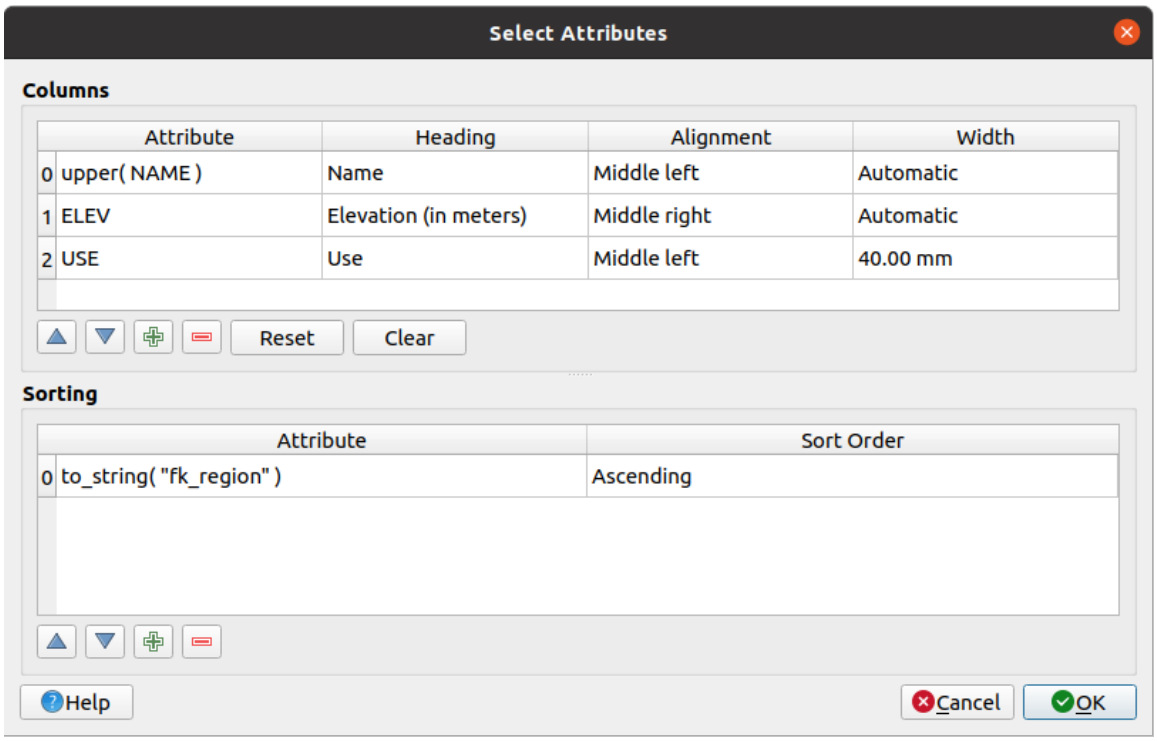


図 21.42: 属性テーブルの属性の選択ダイアログ

カラム セクションでは、以下の設定や操作ができます：

- 行を選択して  や  ボタンを使用して行をシフトさせることで、属性をリスト内で上下に移動させることができます。複数行を選択して同時に移動させることもできます。
-  ボタンで属性の追加ができます。ボタンを押すとテーブルの一番下に空の行が追加され、属性値として使用するフィールドを選択したり、正規表現式を用いて属性値を作成することができます。
-  ボタンで属性の削除ができます。複数行を選択して同時に削除することができます。
- **リセット** ボタンを使用して、属性テーブルをデフォルトの状態にリセットすることができます。
- **クリア** ボタンを使用して、テーブルをクリアすることができます。テーブルは巨大だが、表示したい属性は少数のみのときに使用すると便利です。各行を手動で削除する代わりに、テーブ

ルをクリアしてから必要な行を追加する方が手早いかもしれません。

- **見出し** カラムにカスタムのテキストを追加することで、セル見出しを変更できます。
- セル内の文字の整列は、テーブルのセル内のテキストの位置を決定する **整列** カラムで管理できます。
- セルの幅は、 **幅** カラムにカスタム値を入力することで、手動で管理することができます。

ソーティング セクションでは、以下の設定や操作ができます：

- テーブルをソートするための属性を追加できます： ボタンを押すと、空の行が追加されます。 **属性** カラムにフィールドまたは式を入力し、 **ソート順** を **昇順** または **降順** に設定します。
- リストから行を選び、 や ボタンを使用して、属性レベルでのソートの優先度を変更できます。 **ソート順** カラムのセルを選択すれば、属性フィールドのソート順を変更できます。
- ボタンを使用して、ソーティングリストから属性を削除できます。

地物のフィルタリング

属性テーブルの **地物のフィルタリング** グループには、以下の機能があります（ 図 21.43 参照）：

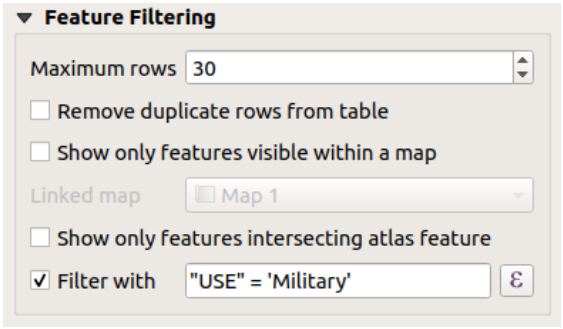


図 21.43: 属性テーブルの地物のフィルタリンググループ

ここでは、以下の設定ができます：

- 表示される **最大行数** を定義できます。
- **テーブルから重複行を削除** を有効にすると、ユニークなレコードのみが表示されます。
- **地図内の可視地物のみを表示する** を有効にし、対応する **リンクされた地図** を選択すると、可視状態の地物の属性が表示されます。
- **地図帳を作成する** が有効になっている場合にのみ、 **地図帳地物と交差する地物のみを表示する** を有効にできます。これを有効にすると、現在の地図帳地物と交差する地物のみがテーブルに表示されます。
- **フィルタ** を有効にして、入力行に入力するか 式ボタンを使用して正規表現を入力することで、フィルタをかけることができます。サンプルデータセットから airports レイヤを読み込んだ場合に使えるフィルタ式の例をいくつか示します：
  - `ELEV > 500`

- `NAME = 'ANIAK'`
- `NAME NOT LIKE 'AN%'`
- `regexp_match( attribute( $currentfeature, 'USE' ) , '[i]')`

最後の正規表現式は、属性フィールド「USE」に文字「i」が含まれる空港だけを抽出しています。

外観

属性テーブルの **外観** グループには、以下の機能があります（ 図 21.44 参照）：

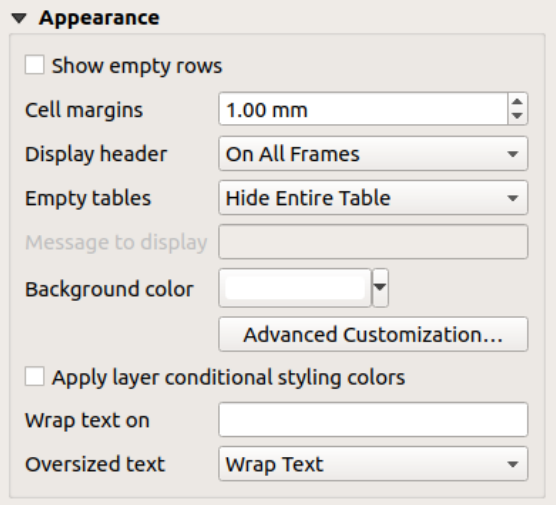


図 21.44: 属性テーブルの外観グループ

- **空行を表示** をチェックすると、属性テーブルが空のセルで埋められます。このオプションは、結果を表示する際に、空のセルを追加で用意するためにも使用できます。
- **セルのマージン（余白）** は、テーブルの各セルでテキストの周りのマージンを定義できます。
- **ヘッダの表示** は、「最初のフレーム」、デフォルトのオプションの「全フレーム」、または「ヘッダなし」をリストから選べます。
- **値の無いテーブル** は、選択結果が空となった場合の表示を制御します。
  - **ヘッダのみ表示** はヘッダのみを表示します。ただし、 **ヘッダの表示** で「ヘッダなし」を選んだ場合は除きます。
  - **表全体を隠す** は、テーブルの背景のみ描画します。 **フレーム** で **フレームのが空の場合、背景を描画しない** を有効にすると、テーブルは完全に非表示となります。
  - **設定メッセージを表示** は、ヘッダを表示し、全カラムの幅のセルを追加して、 **表示するメッセージ** オプションに入力された「結果なし」のようなメッセージを表示します。
- **表示するメッセージ** は、 **値の無いテーブル** で **設定メッセージを表示** を選んだ場合のみ有効となります。 ここに入力されたメッセージは、結果が空のテーブルになった場合の 1 行目に表示されます。
- **背景色** は、 **色セレクタ** ウィジェットを使用してテーブルの背景色を設定できます。 **詳細カスタマイズ** オプションを使用すれば、各セルで異なる背景色を定義することができます（ 図 21.45 参照）。

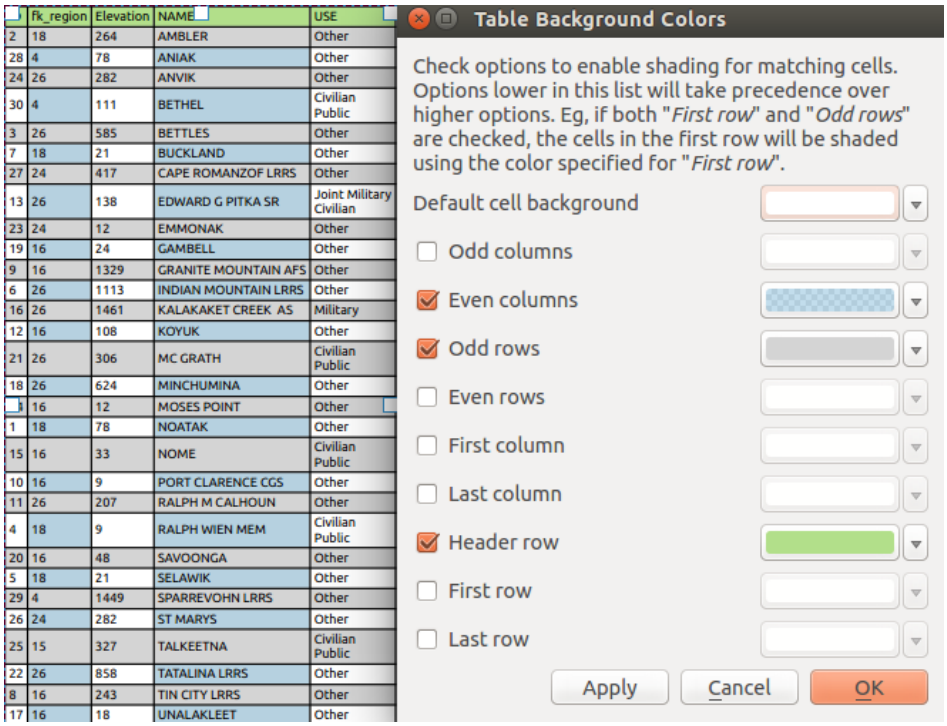


図 21.45: 属性テーブルのテーブルの背景色ダイアログ

- **条件付きスタイルルールを適用**: レイヤにある **条件付きテーブル書式設定** は、レイアウト属性テーブルの内部で適用されます (背景色、フォントファミリ、および太字、斜体、取り消し線、下線、色などのプロパティ)。条件付き書式ルールは他のレイアウトテーブルの書式設定よりも優先されます。例えば、行の色を交互に変えるなど、他のセルの背景色の設定よりも優先されます。
- **テキスト改行場所** オプションでは折り返し文字を定義でき、セルのコンテンツにその文字が現れるたびに文字列が折り返されます。
- **特大テキスト** は、カラムに設定された幅がコンテンツの長さよりも小さい場合振る舞いを定義します。 **テキストの折り返し** または **テキストの切り捨て** を設定できます。

注釈: 属性テーブルアイテムのその他のプロパティは、 **属性テーブル・固定テーブルで共通の機能** のセクションで説明しています。

固定テーブルアイテム

地図に関する追加的な情報は、手作業でテーブルに挿入することができます。 **固定テーブルを追加** を選択して、 **アイテムの作成手順** に従い、新しいテーブルアイテムを追加します。これは作成後、 **レイアウトアイテムの操作** と同じ方法で操作できます。

デフォルトでは、2 行 2 列で最小幅の空のテーブルが地図印刷レイアウトに出現します。テーブルは **アイテムプロパティ** パネルでカスタマイズします。 **アイテムの共通プロパティ** の他に、固定テーブルには以下の機能があります。

メインプロパティ

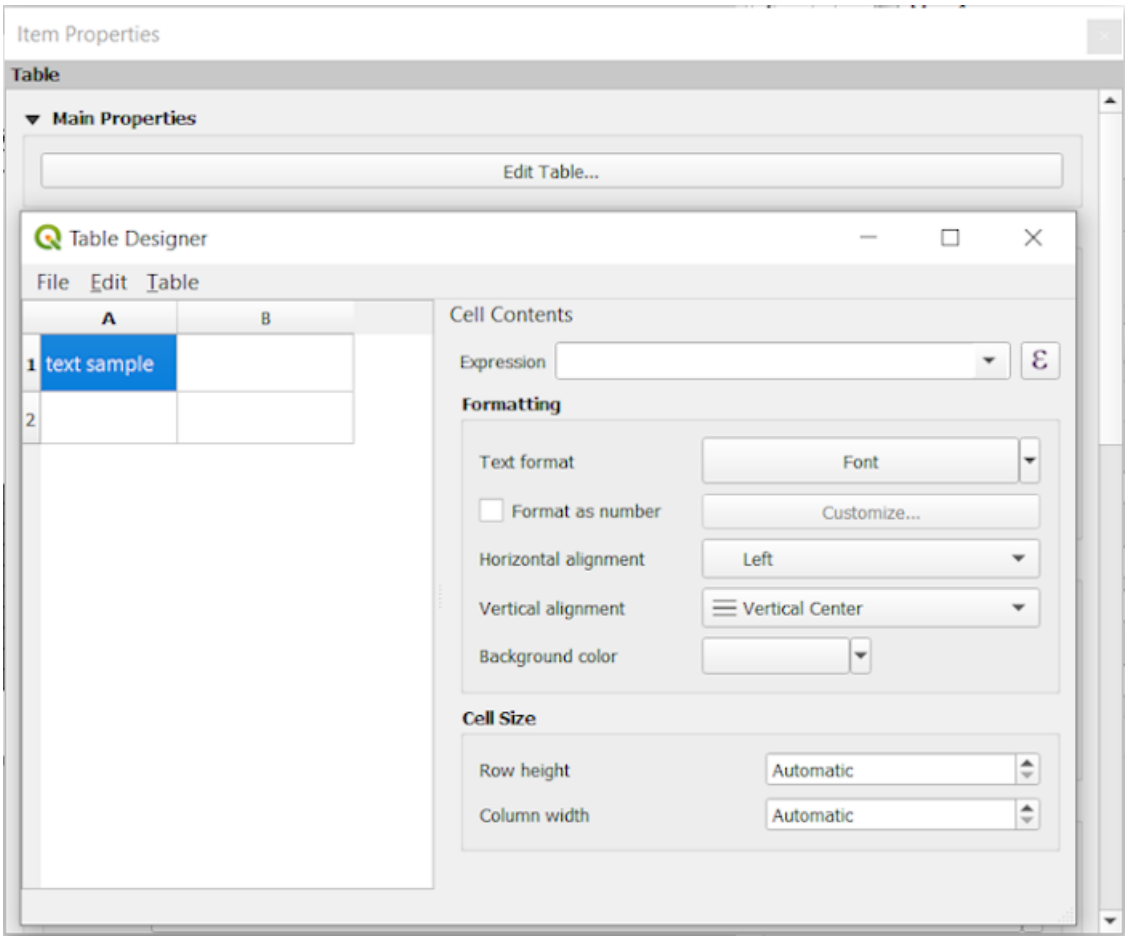


図 21.46: 固定テーブルアイテムのプロパティパネルとテーブルデザイナー

メインプロパティ では、 **テーブルを編集**... をクリックして開く **テーブルデザイナー** で作業を行います。

- テーブル内をクリックし、テキストを手作業で挿入できます。
- 上部にあるメニューから、以下の操作ができます：
  - **ファイル** メニューから、**クリップボードから内容を貼り付ける**（入力されたテーブルを上書きします）
  - **編集** メニューから、行や列の選択機能を使用する
  - **行を挿入** 、 **列を挿入** 、 **行を削除** 、 **列を削除** ならびに **ヘッダ行を含む** オプションの使用
- 右にある **セル内容** セクションでは、以下の設定ができます：
  - **フォーマット** で、選択したセルのテキストフォーマットの定義
    - 式ボタンをクリックして、セルの入力に正規表現を使用します。
    - **テキストフォーマット** を選択します。
    - **番号でフォーマット**（いくつかの数値フォーマットが利用可能です）

    * **水平方向配置** と **垂直方向配置** の定義
    * **背景色** の選択
  - **行の高さ** と **カラム幅** によって **セルサイズ** を定義します。

外観

固定テーブルの **外観** グループには、下記の機能があります：

- **空行を表示** をクリックすると、固定テーブルを空のセルで埋めます。
- **セルのマージン（余白）** は、テーブルの各セルでテキストの周りのマージンを定義できます。
- **ヘッダの表示** は、「最初のフレーム」、デフォルトのオプションの「全フレーム」、または「ヘッダなし」をリストから選べます。
- **背景色** は、**色セレクタ** ウィジェットを使用してテーブルの背景色を設定できます。 **詳細カスタマイズ** オプションを使用すれば、各セルで異なる背景色を定義することができます。
- **特大テキスト** は、カラムに設定された幅がコンテンツの長さよりも小さい場合振る舞いを定義します。 **テキストの折り返し** または **テキストの切り捨て** を設定できます。

注釈: 固定テーブルアイテムのその他のプロパティは、**属性テーブル・固定テーブルで共通の機能** のセクションで説明しています。

属性テーブル・固定テーブルで共通の機能

グリッド表示

テーブルアイテムの **グリッドを表示** グループには、以下の機能があります（ 図 21.47 参照）：

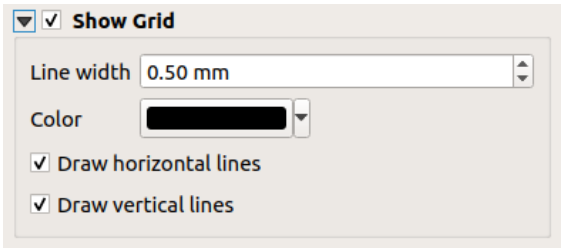


図 21.47: 属性テーブルのグリッドを表示グループ

- テーブルにグリッド（セルの外枠）を表示したい場合には、**グリッドを表示** を有効にします。また、 **水平線を描画** 、 **垂直線の描画** またはその両方を選択できます。
- **線幅** は、グリッドに使用する線の線幅を指定します。
- グリッドの **色** は、色セレクタウィジェットを使用してグリッド色の設定ができます。

フォントとテキストのスタイル

テーブルアイテムの **フォントとテキストのスタイル** グループには、以下の機能があります (図 21.48 参照) :

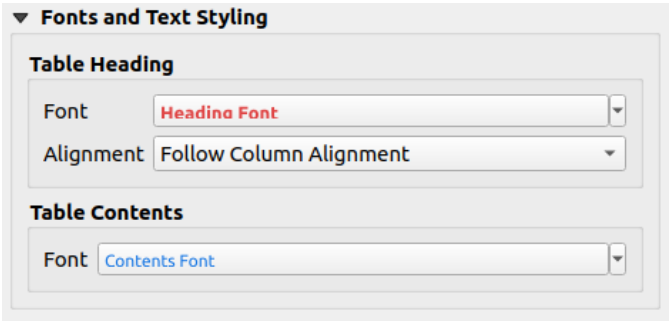


図 21.48: 属性テーブルのフォントとテキストのスタイルグループ

- **テーブルの見出し** と **テーブルの中身** (*contents*) の **フォント** プロパティを、高度な **テキスト形式** ウィジェット (バッファ、影、描画エフェクト、不透明度、背景、色など) を使用して定義することができます。この変更は、**外観** セクションや **テーブルデザイナー** ダイアログによってセルに設定されたカスタムフォントには影響を与えないことに留意してください。デフォルトの設定でレンダリングするセルのみが、スタイル設定を上書きされます。
- **テーブルの見出し** については、さらに **整列** の設定を カラムの配置に従う 、または 左 、 中央 、 右 から選択して設定を上書きすることができます。カラムの整列には、 **属性の選択** ダイアログ (図 21.42 参照) を使用します。

フレーム

テーブルアイテムプロパティの **フレーム** グループには、以下の機能があります (図 21.49 参照) :

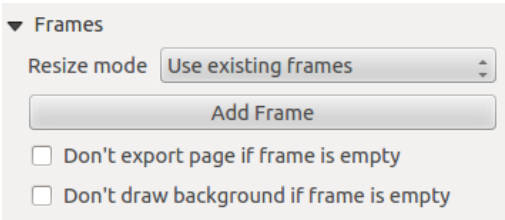


図 21.49: 属性テーブルのフレームグループ

- **リサイズモード** を使用して、属性テーブルのコンテンツをどのようにレンダリングするかを選択できます。
  - 既存フレームを利用 は、最初のフレームと追加されたフレームのみにテーブルを表示します。
  - 次のページへ拡張 は、属性テーブルの選択アイテム全てを表示するのに必要な数のフレーム (とそれに対応するページ) を作成します。各フレームは、レイアウト上で移動させることができます。フレームのサイズを変更した場合には、出来上がったテーブルは別のフレームに分割されます。最後のフレームはテーブルに合わせてトリミングされます。
  - 終了まで繰り返す は、 **次のページへ拡張** オプションと同様に必要なだけフレームを作成しますが、全てのフレームが同じサイズとなります。

- **フレームを追加** ボタンを使用して、選択したフレームと同じサイズのフレームを追加することができます。リサイズモードで 既存フレームを利用 を選んでいる場合には、最初のフレームに入らないテーブルの結果は、次のフレームに続きが表示されます。
- **フレームの内容が無い場合にはページをエクスポートしない** を有効化すると、テーブルフレームのコンテンツが無い場合には、ページはエクスポートされません。これにより、その他すべてのレイアウトアイテム、地図、スケールバー、凡例なども結果に表示されなくなります。
- **フレームが空の場合、背景を描画しない** を有効化すると、テーブルフレームのコンテンツが無い場合には、背景が描画されません。

21.2.8 マーカー、画像、方位記号アイテム

印刷レイアウトで地図や凡例アイテムと一緒に、画像や注釈で出力を装飾したい場合があります。QGIS はこれを実現するための様々なツールを提供しています：

- 画像アイテム: 画像ラスタまたは SVG ファイル（ロゴ、写真、方位記号など）でレイアウトを装飾します。
- **方位記号アイテム**: 方位記号の画像で定義された画像（ピクチャ）アイテム
- **マーカーアイテム**: QGIS のベクタ **シンボル** でレイアウトを装飾します。これは、マップアイテムの上にマーカーを配置したり、高度なカスタム凡例を作成するために使うことができます。

画像アイテム

画像を追加するには、ファイルマネージャからキャンバスにドラッグする、`Ctrl+V` または **編集** **貼り付け** を使ってレイアウトに直接貼り付けるか、画像を追加 を使う *items creation instructions* に従うことでできます。それから **レイアウトアイテムの操作** で説明されているように操作することができます。

画像を追加 を使用すると、画像アイテムは空白のフレームになり、**アイテムのプロパティ** パネルを使用してカスタマイズすることができます。この機能には *items common properties* 以外に以下の機能があります：

メインプロパティ

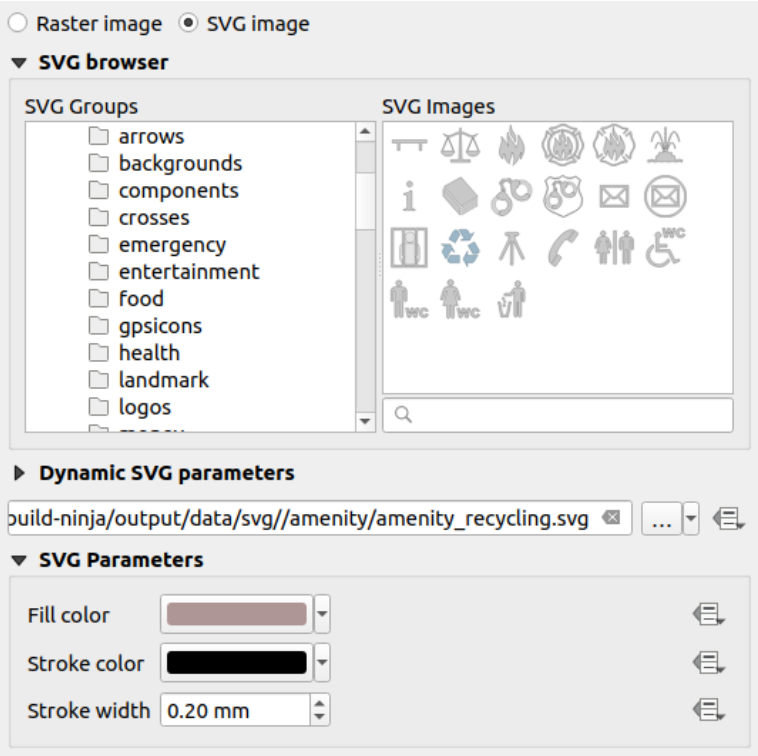


図 21.50: 画像アイテムのアイテムプロパティパネル

画像アイテムは 2 種類の画像をサポートしています。

- **ラスタ画像**: ファイルセレクタウィジェットを使用してデータを取得することができます。guilabel:... ブラウズ ボタンを使ってコンピュータ上のファイルを選択するか、テキストフィールドに直接パスを入力してください。画像を指すリモート URL を指定することもできます。関連する画像はレイアウトに:ref:`埋め込む <embedded_file_selector>`ことできます。

  データによって定義された上書き ボタンを使用して、地物の属性や正規表現を使用して画像のソースを設定することができます。

- *SVG* **イメージ**: **設定** **オプション** **システム** *SVG* **パス** で与えられている SVG ライブラリをデフォルトで使用します。しかし、他のファイルを使用することもでき、ファイル選択はラスタ画像と同じルールに従います。SVG パラメータは動的に設定することもできます。

  QGIS が提供する（デフォルトの） `.SVG` ファイルはカスタマイズが可能です。つまり、 *SVG* **パラメータ** グループ内の対応する機能を使用して、 **塗りつぶし色** や **ストローク色** （不透明度も含む）、**ストローク幅** にオリジナルとは別の値を簡単に適用させることができます。これらのプロパティは **データ定義** とすることもできます。

  追加した `.SVG` ファイルでこれらのプロパティが有効でない場合には、不透明度などをサポートするために、以下のタグを追加する必要があります：

  – *fill-opacity="param(fill-opacity)"*

  – *stroke-opacity="param(outline-opacity)"*

より詳細は次にあります：

注釈: 画像ファイル（ラスタまたは SVG）をレイアウトページにドラッグ＆ドロップすると、対応する設定のレイアウト画像アイテムが作成されます。

サイズと配置

図 21.51: レイアウト画像のサイズと配置プロパティ

リサイズモード オプションでは、フレームの大きさを変更したときに画像がどのように表示されるかを設定できます：

- ズーム ：元のアスペクト比を保ったまま、フレームに合わせて画像を拡大／縮小します。
- 伸ばす ：フレーム内部に合わせるように画像を伸縮させます。
- 切り抜く（`clip`） ：このモードはラスタ画像でのみ使用できます。このモードでは、画像サイズを拡大縮小なしのオリジナルの画像サイズとし、画像はフレームで切り取られます。従って、フレームの内側にある画像の一部のみが表示されます。
- ズームしてフレームをリサイズ ：画像をフレームに合うように拡大し、それから拡大後の画像の大きさに合わせてフレームの大きさを変更します。
- フレームを画像サイズにリサイズ ：フレームのサイズを（拡大縮小なしの）元の画像サイズに合わせるよう設定します。

選択した **リサイズモード** によっては、 **配置** と **画像の回転** オプションが無効になることがあります。 **配置** では、フレーム内の画像の位置（上/中/下、左/中央/右）が選択できます。

画像の回転

画像は **画像の回転** フィールドで回転させることができます。 **地図と同期する** チェックボックスを有効にすると、画像の回転が選択されている地図アイテムに適用されている回転と同期します。これはどんな画像でも方位記号として動作させることができる便利な機能です。**ノースアライメント** には次のような指定ができます:

- **グリッドの北** ：国単位／ローカルグリッドの中央子午線に平行なグリッド線の方向
- **真北** ：経度の子午線の方向

画像の回転には、傾きの **オフセット** を適用することができます。

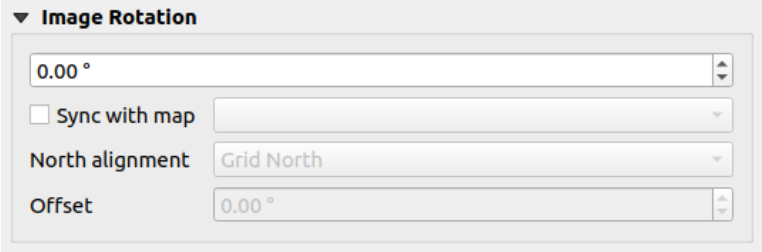


図 21.52: レイアウト画像の画像の回転プロパティ

方位記号アイテム

方位記号を追加 ボタンを使用して、 **アイテムの作成手順** に従えば、方位記号を追加できます。これは、 **レイアウトアイテムの操作** と同じ方法で操作できます。

方位記号が画像であるため、 **方位記号** アイテムは **画像アイテム** と同じプロパティを持っています。主な違いは、

- 空のフレームの代わりに、デフォルトの北向き矢印記号が使用されます。
- 方位記号アイテムは、デフォルトで地図アイテムと同期しています。 **地図と同期する** プロパティは、地図記号アイテムの下にある地図に設定されます。下に地図がない場合には、 **参照マップ** にフォールバックします。

注釈: 方位記号の多くには、北を表す「N」という文字がありません。これは北に「N」を使用しない言語があるため、意図的にそのようになっています。

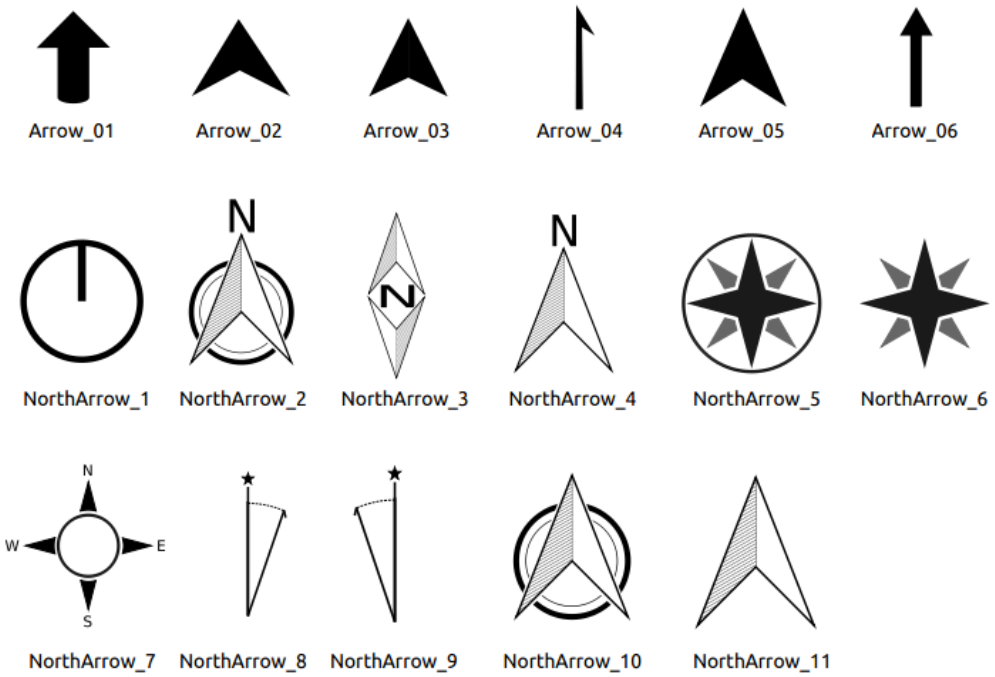
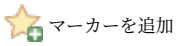


図 21.53: SVG ライブラリで提供されている方位記号

マーカーアイテム

マーカーアイテムを追加するには、 

ボタンを選択し、ページ上でクリックします。デフォルトのポイントマーカーシンボルが追加されます。それから **レイアウトアイテムの操作** で説明したように操作することができます。しかし、他のほとんどのアイテムとは異なり、アイテムのサイズは埋め込まれたシンボルのプロパティによって制御されることに注意してください。

マーカーアイテムは **アイテムプロパティ** パネルからカスタマイズすることができます。ref:**アイテムの共通プロパティ** <*item_common_properties*> 以外にも、以下のことができます：

- シンボルの **ウィジェットの能力** に依存して **シンボル** を変更する
- マーカーアイテムの回転を地図と同期させ (**画像の回転** を参照)、方位記号として動作します。地図の回転は既存のマーカーシンボルレベルの回転に追加されます （例えば、三角形のマーカーを真上に向けるために 90 ° 回転させる必要があっても、北矢印モードではうまく機能します！）

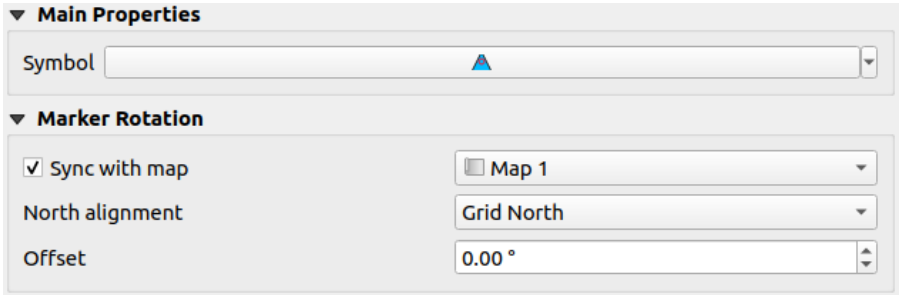


図 21.54: マーカーアイテムのカスタムプロパティ

21.2.9 HTML フレームアイテム

ウェブサイトのコンテンツを表示したり、自身で作成しスタイリングした HTML ページを表示するフレームを追加することができます！ *HTML* **を追加** を使用して、 **アイテムの作成手順** に従うことで HTML を追加でき、これは **レイアウトアイテムの操作** で説明されているものと同様に操作できます。HTML の縮尺は、HTML フレームが作成された時点でのレイアウトのエクスポート解像度によって制御されることに留意してください。

HTML アイテムは **アイテムプロパティ** パネルを使用してカスタマイズすることができます。 **アイテムの共通プロパティ** の他に、HTML アイテムには以下の機能があります（ 図 21.55 参照）。

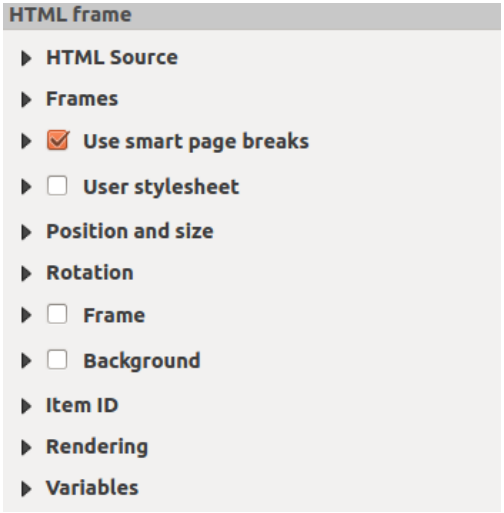


図 21.55: HTML フレームのアイテムプロパティパネル

HTML ソース

HTML フレームの **アイテムプロパティ** パネルの *HTML* **ソース** グループには、下記の機能があります（ 図 21.56 参照）：

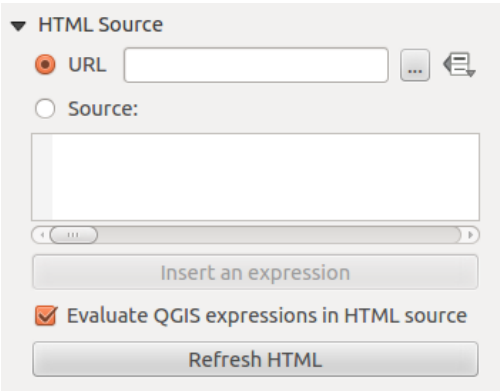


図 21.56: HTML フレームの HTML ソースプロパティ

- *URL* では、インターネットブラウザからコピーしたウェブページの URL の入力や、 ... ブラウズ ボタンを使用して HTML ファイルの選択ができます。 データによって定義された上書き ボタンを使用するオ

プションもあり、テーブルの属性フィールドのコンテンツや、正規表現を使用して URL を指定することができます。

- **ソース** には、テキストボックスに HTML タグ付きのテキストを入力したり、完全な HTML ページを入力することができます。
- **式の挿入・編集**... ボタンを使用すると、「ソース」テキストボックスに `[%Year($now)%]` 式を追加して、現在の年を表示させるようなことができます。このボタンは、ラジオボタンの **ソース** が選択されている場合のみ有効です。式を入力したら、HTML フレームが更新される前にテキストボックス内のどこかをクリックしてください。そうしないと、式が失われてしまいます。
- *HTML* **ソース内の** *QGIS* **式の評価** にチェックを入れると、ソース内に含まれる QGIS 式の評価結果が表示されます。チェックを入れない場合には、結果の代わりに式そのものが表示されます。
- *HTML* **のリフレッシュ** ボタンを使用すると、HTML フレームが更新され、変更した結果が表示されます。

フレーム

HTML フレームの **アイテムプロパティ** パネルの **フレーム** グループには、下記の機能があります（図 21.57 参照）：

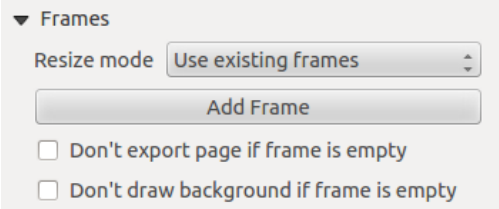


図 21.57: HTML フレームのフレームプロパティ

- **リサイズモード** を使用して、HTML コンテンツをどのようにレンダリングするかを選択できます。
  - 既存フレームを利用 は、最初のフレームと追加されたフレームのみにレンダリング結果を表示します。
  - 次のページへ拡張 は、ウェブページの長さを表示するのに必要な数のフレーム（とそれに対応するページ）を作成します。各フレームは、レイアウト上で移動させることができます。フレームのサイズを変更した場合には、ウェブページは別のフレームに分割されます。最後のフレームはウェブページに合わせてトリミングされます。
  - 各ページで繰り返す は、各ページで同じサイズのフレームで、ウェブページの左上部分を繰り返しレンダリングします。
  - 終了まで繰り返す は、次のページへ拡張 オプションと同様に必要なだけフレームを作成しますが、全てのフレームが同じサイズとなります。
- **フレームを追加** ボタンを使用して、選択したフレームと同じサイズのフレームを追加できます。 **リサイズモード** で **既存フレームを利用** を使用している場合には、HTML ページが最初のフレームに収まらない場合、次のフレームに続きが表示されます。

- **フレームの内容が無い場合はページをエクスポートしない** を有効化すると、HTML コンテンツが無い場合には、ページはエクスポートされません。これにより、その他すべてのレイアウトアイテム、地図、スケールバー、凡例なども結果に表示されなくなります。
- **フレームが空の場合、背景を描画しない** を有効化すると、HTML フレームが空の場合には、HTML フレームが描画されません。

スマートページブレークとユーザスタイルシートの使用

HTML フレームの **アイテムプロパティ** パネルの *Smart Page Breaks* **を使う** ダイアログと **スタイルシートを使用する** ダイアログには、下記の機能があります（ 図 21.58 参照）：

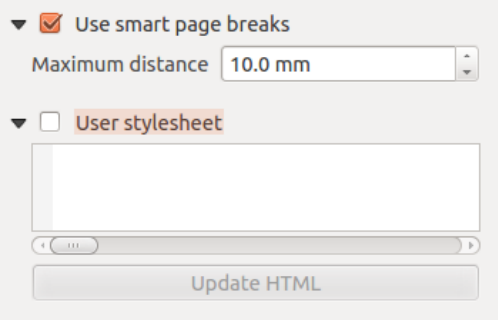


図 21.58: HTML フレームの「Smart Page Breaks を使う」プロパティと「スタイルシートを使用する」プロパティ

- *Smart Page Breaks* **を使う** にチェックを入れると、HTML フレームコンテンツがテキスト行の途中で改ページされるのを防ぎ、次のフレームにスムーズに続くようにします。
- HTML のどこで改ページを行うかを計算する際に許容される **最大距離** を設定します。この距離は、最適な改ページ位置の計算後に、フレームの下部に許される空白の最大量です。より大きな値を設定すると、改ページ位置の選択が柔軟になされますが、フレームの下部に無駄な空白ができやすくなります。これは、 *Smart Page Breaks* **を使う** が有効な場合にのみ使用できます。
- **スタイルシートを使用する** を有効にすると、CSS で指定される HTML スタイルを適用することができます。以下に、スタイルコードの例を示します。 <h1> ヘッダタグの色を緑に設定し、パラグラフタグ <p> に含まれるテキストのフォントとフォントサイズを設定するものです。

```
h1 {color: #00ff00;
}
p {font-family: "Times New Roman", Times, serif;
   font-size: 20px;
}
```

- *HTML* **の更新** ボタンを使用して、スタイルシートの設定結果を確認できます。

21.2.10 Shape アイテム

QGIS には印刷レイアウト上に基本的な図形や複雑な図形を描くためのツールがいくつかあります。

注釈: 他の印刷レイアウトアイテムとは異なり、図形の枠や背景色（デフォルトで透明の設定）のスタイルを設定することはできません。

基本的な Shape アイテム

Shape アイテムは、三角形や四角形、楕円といった基本的な図形で地図を装飾するために用いるツールです。 Shape を追加 ツールを使用して、基本的な図形を追加することができます。このツールは、 四角形を追加 、 楕円を追加 、 三角形を追加 といった特定のツールへのアクセスを提供します。適切なツールを選択したら、 **アイテムの作成手順** に従ってアイテムを描画します。他のレイアウトアイテムと同様、基本図形アイテムは **レイアウトアイテムの操作** と同じ方法で操作できます。

注釈: `Shift` キーを押しながらクリック＆ドラッグで基本図形を描画すると、正方形や円、正三角形を作成できます。

デフォルトの Shape アイテムは、 **アイテムプロパティ** パネルを使用してカスタマイズすることができます。 **アイテムの共通プロパティ** の他に、Shape アイテムには以下の機能があります（ 図 21.59 参照）。

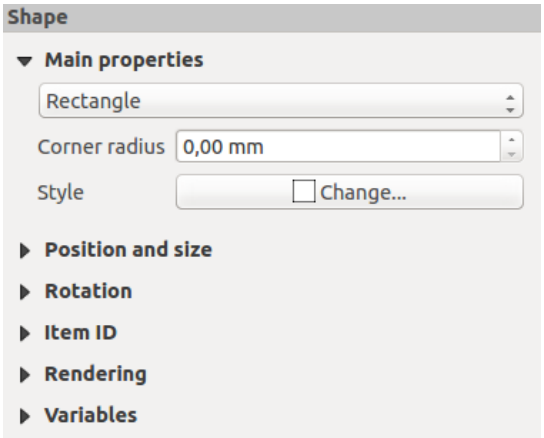


図 21.59: Shape アイテムプロパティパネル

メインプロパティ グループには Shape アイテムの種類（ **楕円** 、 **四角形** または **三角形** ）が表示され、フレーム内で種類を切り替えられます。

高度な **シンボル** と **色** のセレクタウィジェットを使用して、図形のスタイルを設定できます。

四角形については、 **コーナー半径** の値を様々な単位で設定することで、角を丸められます。

ノードに基づく図形アイテム

Shape **を追加** ツールを使えば、シンプルな既定の幾何学的アイテム作成できますが、**ノードアイテムを追加** ツールを使えば、カスタムのより複雑な幾何学アイテムを作成することができます。ポリラインやポリゴンとして、好きなだけ線分または辺を作成でき、アイテムの頂点は **ノードアイテムの編集** を使用して個別に直接操作することができます。このアイテム自体は **レイアウトアイテムの操作** と同様に操作できます。

ノードに基づく図形を追加するには：

1. ノードアイテムを追加 アイコンをクリックします
2. ポリゴン（Polygon）を追加 または ポリラインを追加 ツールを選択します
3. 一連の左クリックにより、アイテムのノードを追加します。 `Shift` キーを押しながらセグメントを描画すると、 45 ° の倍数に方向が制限されます。
4. ノードを追加し終えたら、右クリックして図形描画を終了します。

アイテムプロパティ パネルで図形の見た目をカスタマイズできます。

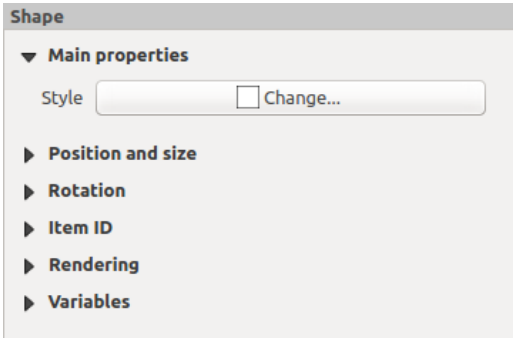


図 21.60: ポリゴンノード図形のアイテムプロパティパネル

メインプロパティ では、高度な **シンボル** と **色** のセレクタウィジェットを使用して、図形のスタイルを設定できます。

ポリラインノードアイテムに対しては、 **ラインマーカー** の変更ができます。すなわち、

- 開始／終了マーカーには以下のオプションがあります。
  - **なし** ：単なるポリラインを描画します
  - **矢印** ：カスタマイズも可能な、通常の三角形の矢印の頭を追加します
  - *SVG* マーカー： SVG ファイルをアイテムの矢印の頭として使用します
- 矢印のカスタマイズとしては、
  - **矢印のストローク色** ：矢印の頭のストローク色を設定します
  - **矢印の塗りつぶし色** ：矢印の頭の塗りつぶし色を設定します
  - **矢印ストローク幅** ：矢印の頭のストローク幅を設定します

– **矢印ヘッド幅** ：矢印の頭のサイズを設定します

SVG 画像の場合はラインに合わせて自動的に回転します。QGIS の既定の SVG 画像は、ストロークと塗りつぶし色を対応するオプションで変更できます。カスタムの SVG では、いくつかのタグが必要となるため、この **説明** に従ってください。

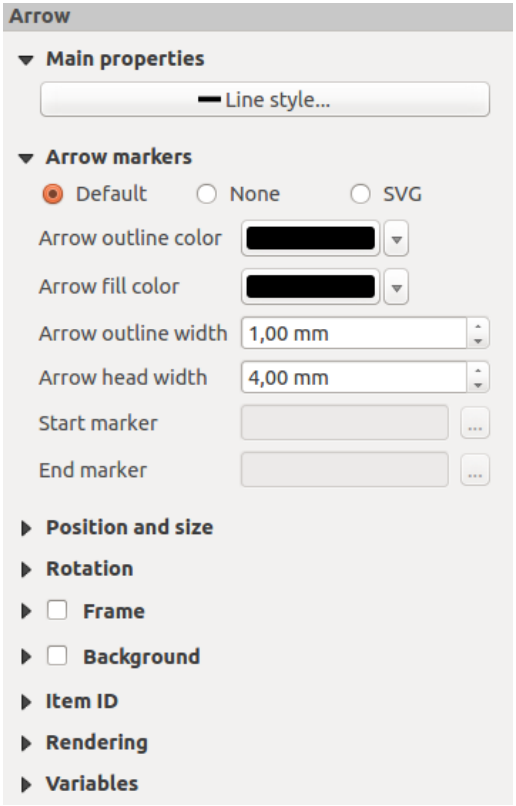


図 21.61: ポリラインノード図形のアイテムプロパティパネル

矢印アイテム

矢印を追加 ツールは、デフォルトで矢印ありとしたポリラインを作成するためのショートカットです。従って、 **ポリラインノードアイテム** と同様のプロパティと動作を持ちます。

実際、矢印アイテムは単純な矢印を追加するために使用できます。例えば、2 つの異なる印刷レイアウトアイテムの関係を示すような使い方です。ただし、北向きの矢印を作成する場合には、 **画像アイテム** の利用をまずは考えてみてください。なぜならば、画像アイテムでは .SVG 形式の北向き矢印のセットにアクセスでき、地図アイテムと同期して自動的に回転させることができるからです。

ノードアイテムのジオメトリ編集

ノードベースの図形を編集するための特別なツールとして、ノードアイテムの編集 ツールが用意されています。このモードでは、ノード上をクリックして選択できます（選択したノードにはマーカーが表示されます)。選択したノードは、ドラッグするか矢印キーを使用して移動できます。さらに、このモードでは既存の図形にノードを追加することもできます。セグメント上をダブルクリックすると、その場所にノードが追加されます。最後に、現在選択しているノードを削除するには、 Del キーを押します。

21.3 出力の作成

図 21.62 に、これまでのセクションで説明したあらゆる種類のレイアウトアイテムを含む印刷レイアウトの例を示します。

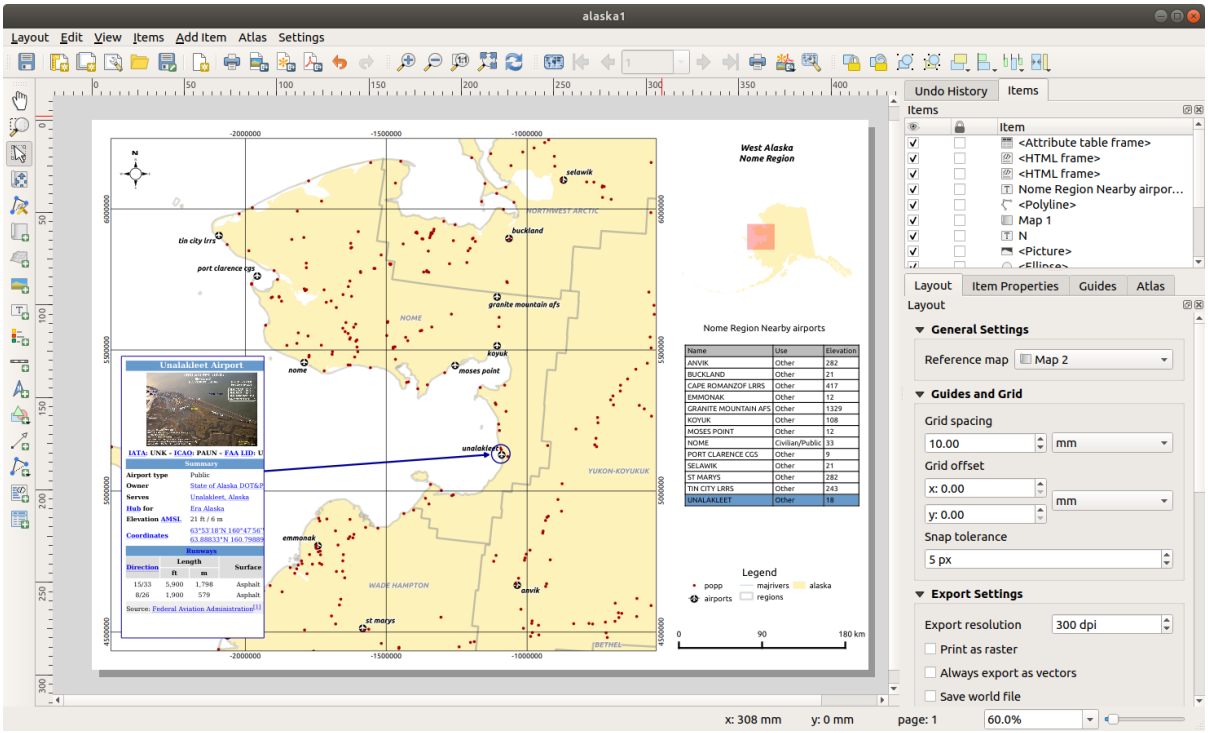


図 21.62: 地図ビュー、凡例、画像、スケールバー、地図グリッド、ラベルテキスト、HTML フレームを追加した印刷レイアウト

レイアウト メニューやツールバーから、印刷レイアウトをさまざまなファイル形式に出力することができ、解像度（印刷の品質）や用紙サイズの変更も可能です。

- 印刷 アイコンは、インストールされているプリンタドライバに応じて、印刷レイアウトを接続されたプリンタで印刷したり、PostScript ファイルに出力できます。
- 画像としてエクスポート アイコンは、印刷レイアウトを PNG 、 BMP 、 TIF 、 JPG やその他の画像形式で出力します。
- SVG としてエクスポート アイコンは、SVG (Scalable Vector Graphic) として印刷レイアウトを保存します。

-  PDF としてエクスポート アイコンは、作成した印刷レイアウトを直接 PDF (Portable Document Format) ファイルとして保存します。

21.3.1 エクスポート設定

印刷レイアウトをエクスポートする際には、最適な出力を行うために QGIS がチェックするエクスポート設定があります。これらの設定は、以下の場所で行います：

- **レイアウト** パネルの **エクスポート設定** における、 **エクスポート解像度** 、 **ラスタとして印刷する** 、 **常にベクタとしてエクスポートする** 、 **ワールドファイルを保存する**
- **ページアイテムプロパティ** パネルにおける、 **ページをエクスポートから除外する**
- **アイテムプロパティ** パネルにおける、 **アイテムをエクスポートから除外する**

さらに、多くの定義済みチェックが自動的にレイアウトに適用されます。現在のところ、これらのチェックには、スケールバーがマップアイテムに正しくリンクされているか、マップの概要アイテムもマップに正しくリンクされているかのテストが含まれます。チェックに失敗した場合は、問題を知らせる警告が表示されます。

21.3.2 画像としてエクスポート

印刷レイアウトを画像としてエクスポートするには：

1.  画像としてエクスポート アイコンをクリックします
2. 使用する画像形式と保存先フォルダ、ファイル名（例 `myill.png` ）を選択します。レイアウトに複数ページが含まれる場合には、指定されたファイル名にページ番号が追加された名前（例 `myill_2.png` ）のファイルに各ページがエクスポートされます。
3. 次に開くダイアログ（ **画像エクスポートオプション** ）では、以下の設定ができます：
   - 印刷レイアウトの **エクスポート解像度** とエクスポートするページの寸法（ **レイアウト** パネルで設定された値）を上書きできます。
   - **アンチエイリアスを有効にする** オプションを有効にすることで、画像のレンダリングを改善できます。
   - レイアウトを **ジオリファレンスされた画像** としてエクスポートしたい場合（例：その他のプロジェクトと画像を共有したい場合）には、 **ワールドファイルを生成する** オプションにチェックを入れると、エクスポートする画像と同名で異なる拡張子（TIFF は `.tfw` 、PNG は `.pnw` 、JPEG は `jgw` など）の *ESRI* **ワールドファイル** がエクスポート時に生成されます。このオプションは、 **レイアウトパネル** の設定でデフォルトでチェック済みにすることもできます。

     注釈: 複数ページの出力では、（ **ワールドファイルを生成する** オプションにチェックが入っている場合、） **参照マップ** が含まれるページにのみワールドファイルが作成されます。

- **内容に合わせて切り取る** オプションにチェックを入れると、レイアウトの各ページで、レイアウトによる出力画像は全てのレイアウトアイテム（地図、凡例、スケールバー、図形、ラベル、画像など）を含んだ最小領域となります。
  - 印刷レイアウトが単一ページの場合には、出力結果は印刷レイアウト上のアイテム全てを含むようにリサイズされます。その後、全てのアイテムの位置（ページ内またはページの上下左右）に応じて、ページは縮小または拡大されます。
  - 複数ページの印刷レイアウトの場合には、領域内にアイテムが収まるように各ページがリサイズされます（全ページの左右と、最初のページの上部、最後のページの下部）。リサイズされた各ページは、それぞれ別のファイルにエクスポートされます。

  内容に合わせて切り取る ダイアログでは、切り取られた境界の周囲にマージンを追加できます。

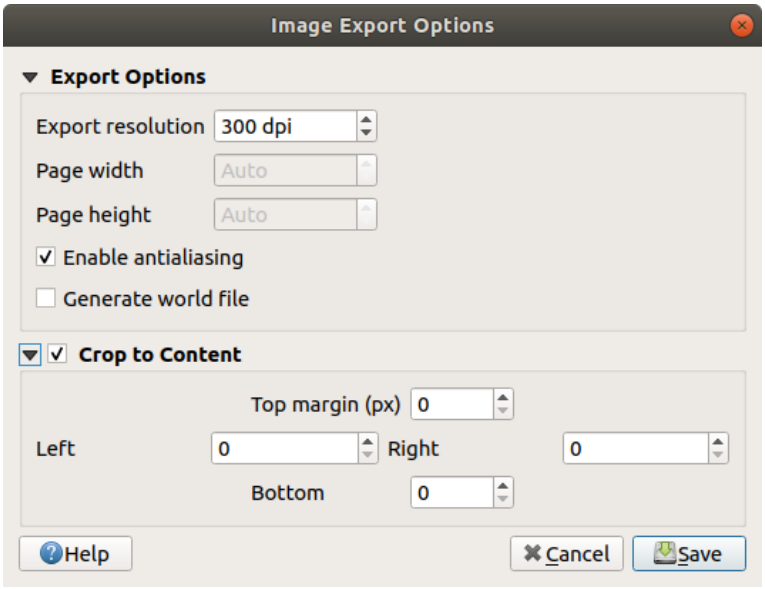


図 21.63: 画像エクスポートオプションで出力結果をアイテムの範囲にリサイズする

Tip: レイアウトアイテムが用紙の範囲をはみ出す場合には、透過をサポートする画像形式を使用してください

レイアウトアイテムは用紙の範囲の外にも配置できます。 **内容に合わせて切り取る** オプションを使用してエクスポートする場合、これによって結果の画像は用紙の範囲をはみ出すことがあります。用紙の範囲の外側の背景は透明であるため、透過をサポートしていない画像形式（例： BMP や JPG ）では、透明な背景は真っ黒でレンダリングされ、画像が「壊れた」ようになってしまいます。このようなケースでは、透過に対応した画像形式（例： TIFF や PNG ）を使用してください。

注釈: 画像形式（例 PNG ）が対応しており、ベースの Qt ライブラリで出力をサポートしている場合、エクスポートされた画像に **プロジェクトのメタデータ** （著者、タイトル、作成日、要約など）を含めることができます。

21.3.3 SVG としてエクスポート

印刷レイアウトを SVG としてエクスポートするには：

1.  SVG としてエクスポート アイコンをクリックします
2. パスとファイル名（画像エクスポートと同様、複数ページのレイアウトの場合にはベース名として使用されるファイル名）を入力します
3. 次に開く *SVG* **エクスポートオプション** ダイアログで、レイアウトのデフォルトの **エクスポート設定** を上書きしたり、新たに設定を行うことができます：
   - ☐ *SVG* **グループとして地図レイヤをエクスポートする** ：エクスポートされるアイテムが QGIS のレイヤ名と同名のレイヤでグループ化されることで、ドキュメントのコンテンツが理解しやすくなります。
   - ☐ **常にベクタとしてエクスポートする** ：一部のレンダリングオプションでは、きれいなレンダリングのためにアイテムをラスタ化する必要があります。このオプションにチェックを入れると、出力ファイルの見た目は印刷レイアウトのプレビューと一致しないリスクはありますが、オブジェクトをベクタとして保持します（詳細については **エクスポート設定** を参照）。
   - ☑ *RDF* **メタデータのエクスポート（***title, author* **など）**：タイトル、著者、日付、説明などをエクスポートします。
   - ☑ **ジオメトリを簡略化してファイルを縮小する** ：このオプションは、ジオメトリ頂点を全てはエクスポートしないようにします。ジオメトリ頂点を全てエクスポートしてしまうと、とてつもなく複雑で巨大なエクスポートファイルサイズになってしまい、他のアプリケーションでは開くことができなくなることがあります。ジオメトリはレイアウトをエクスポートする際に簡略化され、エクスポート解像度では見た目に区別できないような冗長な頂点が削除されます（例えば、エクスポート解像度が `300 dpi` の場合には、`1/600` インチ 未満の間隔の頂点が削除されます）。
   - **テキスト出力** の設定：これは、テキストラベルを適切なテキストオブジェクトとして出力（**テキストを常にテキストオブジェクトとして出力**）するか、パスのみとして出力（**テキストを常にパスとして出力**）するかを制御します。テキストオブジェクトとして出力する場合、外部アプリケーション（Inkscape など）で通常のテキストとして編集が可能です。ただし、副作用としてレンダリング品質が低下し、さらにテキストにバッファ等の特定の設定がなされていると、レンダリングに問題が発生します。このため、テキストをパスとして出力することを推奨します。
   - ☑ **内容に合わせて切り取る** **オプション** の適用
   - ☐ **ラスタタイルのエクスポートを無効化** ：ファイルをエクスポートする際、QGIS は使用メモリの削減のためにラスタレイヤに組み込みのタイルレンダリングを使用します。場合によっては、これが生成されたファイルのラスタに目に見える「継ぎ目」を生じさせる場合があります。このオプションにチェックを入れると、エクスポート中に多量のメモリを消費しますが、この問題を修正できます。

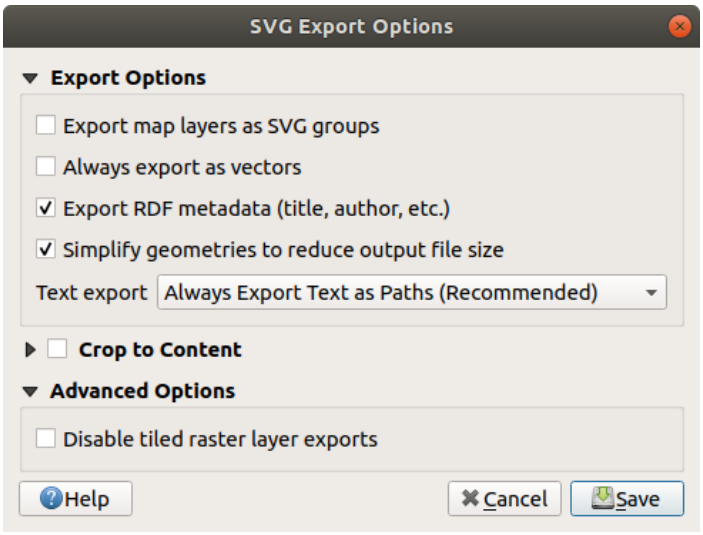


図 21.64: SVG エクスポートオプション

注釈: 現在のところ、SVG 出力は非常に基本的な機能しかありません。これは QGIS の問題ではなく、Qt ライブラリに起因しています。この問題は、将来のバージョンで解決されることを期待しています。

21.3.4 PDF としてエクスポート

印刷レイアウトを PDF としてエクスポートするには：

1. PDF としてエクスポート アイコンをクリックします
2. パスとファイル名を入力します。画像や SVG としてエクスポートする場合とは異なり、レイアウトの全ページが単一の PDF ファイルにエクスポートされます。
3. 次に開く *PDF* **出力のオプション** ダイアログで、レイアウトのデフォルトの **エクスポート設定** を上書きしたり、新たに設定を行うことができます：
   - **常にベクタとしてエクスポートする** ：一部のレンダリングオプションでは、きれいなレンダリングのためにアイテムをラスタ化する必要があります。このオプションにチェックを入れると、出力ファイルの見た目は印刷レイアウトのプレビューと一致しないリスクはありますが、オブジェクトをベクタとして保持します（詳細については **エクスポート設定** を参照）。
   - **地理参照情報を追加** ：情報を取得する **参照マップ** が最初のページにある場合にのみ、利用可能です。
   - *RDF* **メタデータのエクスポート（***title, author* **など）**：タイトル、著者、日付、説明などをエクスポートします。
   - **テキスト出力** の設定：これは、テキストラベルを適切なテキストオブジェクトとして出力（**テキストを常にテキストオブジェクトとして出力**）するか、パスのみとして出力（**テキストを常にパスとして出力**）するかを制御します。テキストオブジェクトとして出力する場合、外部アプリケーション（Inkscape など）で通常のテキストとして編集が可能です。ただし、副作用とし

てレンダリング品質が低下し、さらにテキストにバッファ等の特定の設定がなされていると、レンダリングに問題が発生します。このため、テキストをパスとして出力することを推奨します。

- 以下の設定を使用して PDF の **画像圧縮** について制御できます：
  - *Lossy (JPEG)* これはデフォルトの圧縮モードです
  - *Lossless*, 多くの場合はより大きなファイルを生成しますが、出力を印刷したり、外部アプリケーションで後処理を行う場合には適したオプションです（Qt 5.13 以降が必要）。
- **ジオ *PDF (GeoPDF)* を作成**: ジオリファレンスされた PDF ファイルを作成します。
- **ラスタタイルのエクスポートを無効化** ：ファイルをエクスポートする際、QGIS は使用メモリの削減のためにタイルベースのレンダリングを使用します。場合によっては、これが生成されたファイルのラスタに目に見える「継ぎ目」を生じさせる場合があります。このオプションにチェックを入れると、エクスポート中に多量のメモリを消費しますが、この問題を修正できます。
- **ジオメトリを簡略化してファイルを縮小する**: レイアウトのエクスポート中に、エクスポート先の解像度では区別することができない頂点を削除することでジオメトリが簡略化されます（たとえば、エクスポート先の解像度が `300 dpi` ならば、`1/600` インチ よりも近い頂点は削除されます）。これにより、出力ファイルのサイズと複雑さを減少させることができます（非常に大きなファイルは他のアプリケーションでの読み込みに失敗する可能性があります）。

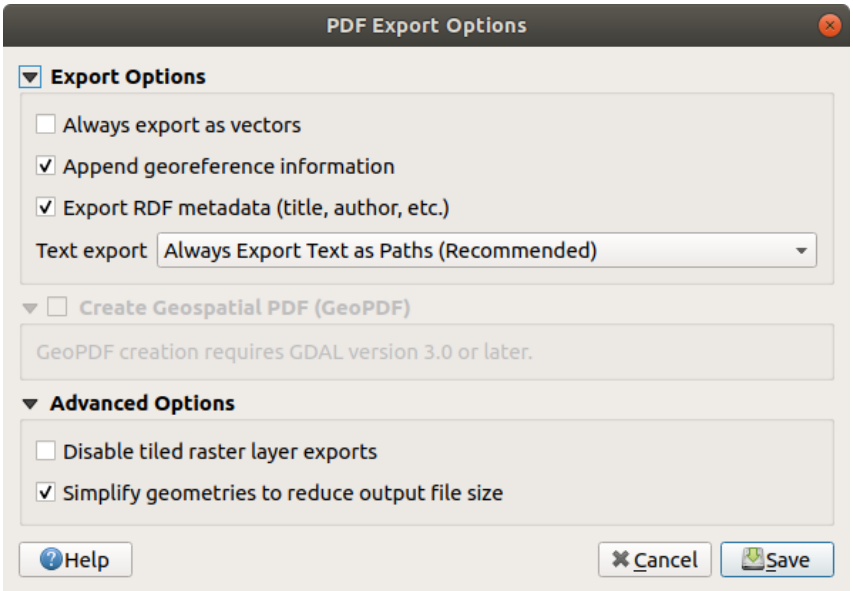


図 21.65: PDF 出力のオプション

注釈: GeoPDF のエクスポートがサポートされており、いくつかの GeoPDF 特有のオプションが利用可能です：

- **形式** （GeoPDF の形式ー GeoPDF にはいくつかのバリエーションがあります）
- **複数の地図テーマを含める** （含める地図テーマを指定します）
- **ベクタ地物情報を含める** （レイヤを選択し、レイヤをを PDF の論理グループにグループ化します）

注釈: 印刷レイアウトを地理参照をサポートする形式（例： PDF や TIFF ）へエクスポートする場合には、デフォルトで地理参照されたアウトプットを生成します。

21.3.5 地図帳の作成

地図帳の機能を使うと、自動で地図帳を作成することができます。地図帳はテーブルやベクタレイヤ（ **カバレッジレイヤ** ）の地物を利用して、テーブル/レイヤの地物（ **地図帳地物** ）毎の出力を作成します。最も一般的な使い方は、現在の地図帳地物に地図アイテムをズームすることです。さらに、以下のような使い方もあります：

- 地図帳地物と別のレイヤについて、地図帳地物と同じ属性値を持つ地物や、地図帳地物のジオメトリ内部にある地物のみを表示した地図アイテムの作成
- 地図帳地物のイテレーションに応じてテキストが置き換えられるラベルや、HTML アイテムの作成
- 現在の地図帳地物に関連した **親または子** 地物の属性を表示したテーブルアイテムの作成

各地図帳地物について、エクスポート設定に応じてすべてのページとアイテムの出力が実行されます。

Tip: 変数を使用したより柔軟な対応

QGIS には、地図帳に関連したものも含め多数の関数や **変数** が用意されています。これにより、地図帳の状態に応じて、レイアウトアイテムだけでなくレイヤのシンボロジも操作することができます。これらの機能を組み合わせることにより、より柔軟な対応が可能となり、高度な地図を簡単に作成することができます。

地図帳の作成を有効にし、地図帳パラメータにアクセスするには、 **地図帳** パネルを開いてください。このパネルには以下の設定があります（ 図 21.66 参照）：

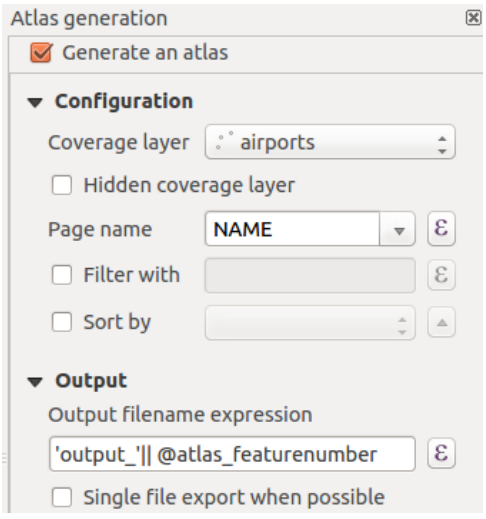


図 21.66: 地図帳パネル

- **地図帳を作成する** は、地図帳の作成を有効または無効にできます。
- **設定**
  - **カバレッジ・レイヤ** コンボボックスでは、イテレーションする地物を含むテーブルまたはベクタレイヤを選択します。
  - オプションの **カバレッジレイヤを隠す** にチェックを入れると、地図帳の作成中はカバレッジレイヤを表示しません（その他のレイヤは表示されます）。
  - オプションの **ページ名称** コンボボックスは、レイヤ地物ページの名前を指定します。カバレッジレイヤのフィールドを選択するか、 **式** を設定できます。このオプションが空欄の場合には、QGIS はレイヤに適用されたフィルタやソートに応じた内部 ID を使用します。
  - オプションの **フィルタ** テキストエリアでは、カバレッジレイヤから地物をフィルタリングするための式を指定することができます。式が空欄の場合には、評価値が `True` となる地物のみが使用されます。
  - オプションの **並べ方** は、カバレッジレイヤのフィールドまたは式を使用して、カバレッジレイヤの地物（と出力結果）を並べ替えます。ソートの順序（昇順または降順）は、上向きまたは下向き矢印が表示された切り替え式の **並べ順** ボタンによって設定できます。
- **出力** - ここでは、地図帳の出力に関する設定ができます：
  - **出力ファイル名の式** テキストボックスは、地図帳地物それぞれに対するファイル名を生成するために使用される式です。これは、式に基づいており、複数のファイルをレンダリングする場合のみ意味があります。
  - **可能であれば一つのファイルに出力** は、選択した出力形式で可能な場合（例えば PDF ）には、単一のファイルを生成するように強制します。このフィールドにチェックが入っている場合には、 **出力ファイル名の式** フィールドの値には意味がありません。
  - **画像のエクスポート形式** ドロップダウンリストでは、 画像として地図帳をエクスポート... ボタンを使用した際の出力形式を選択します。

地図帳による制御

地図帳の最も一般的な使い方は、カバレッジレイヤのイテレーションに従って、現在の地図帳地物に地図アイテムをズームすることです。この動作は、地図アイテムの **地図帳による制御** グループのプロパティで設定を行います。地図アイテムに適用可能なさまざまな設定については、 **地図帳による制御** を参照してください。

式を使ったラベルのカスタマイズ

地図帳がイテレーションする地物に合わせてラベルを変化させるために、ラベルに式を含めることができます。式の部分（関数、フィールド、変数を含む）は必ず [% と %] の間に配置してください（詳細は **ラベルアイテム** を参照してください）。

例えば、CITY_NAME フィールドと ZIPCODE フィールドを持つ city レイヤの場合、次のように入力できます：

```
The area of [% concat( upper(CITY_NAME), ',', ZIPCODE, ' is ',
format_number($area/1000000, 2) ) %] km2
```

または、別の組み合わせの方法として、次のようにもできます：

```
The area of [% upper(CITY_NAME)%],[%ZIPCODE%] is
[%format_number($area/1000000,2) %] km2
```

情報 [% concat(upper(CITY_NAME), ',', ZIPCODE, ' is ', format_number($area/1000000, 2)) %] は、ラベルの内部で使用される式です。どちらの式も、生成される地図帳に以下のような形式のラベルを作成します:

```
The area of PARIS,75001 is 1.94 km2
```

地図帳でのデータによって定義された上書きの利用

印刷レイアウトには、 データによって定義された上書き ボタンを使用して、選択した設定を上書きできる箇所がいくつかあります。これは、地図帳の作成において特に便利です。このウィジェットの詳細については、**データによって定義された上書きの設定** を参照してください。

以下の例では、QGIS サンプルデータセットの Regions レイヤを使用し、これを地図帳作成のための **カバレッジレイヤ** として選択しています。レイアウトは単一ページで、地図アイテムとラベルアイテムを含んでいることを想定しています。

領域範囲の高さ（南北方向）が幅（東西方向）よりも大きい場合には、紙面を有効利用するために **横** 向きではなく、**縦** 向きを使用したいとしましょう。 データによって定義された上書き ボタンを使用すると、用紙の向きを動的に設定することができます。

ページ上を右クリックして、**ページのプロパティ** を選択し、パネルを開きます。領域のジオメトリに応じた式を使って用紙の向きを動的に設定したいので、**方向** フィールドの ボタンを押し、**編集**... を選択して、**式文字列ビルダ** ダイアログを開き、以下の式を入力します。

```
CASE WHEN bounds_width(@atlas_geometry) > bounds_height(@atlas_geometry)
THEN 'Landscape' ELSE 'Portrait' END
```

ここで **地図帳のプレビュー** を行うと、用紙自体は自動的に向きが変わるものの、アイテムの配置位置は望んだものにはなっていないかもしれません。各領域について、レイアウトアイテムの位置も変更する必要があります。地図アイテムの場合は、**幅** プロパティの ボタンを使用して、以下のような式で幅を動的に設定することができます。

```
@layout_pagewidth - 20
```

同様にして **高さ** プロパティの ボタンを使用して以下の式を指定すれば、地図アイテムのサイズを制限することができます。

```
@layout_pageheight - 20
```

地図アイテムがページ内の中心にあるようにするには、**基準点** をラジオボタンの左上に設定して、*X* と *Y* の位置として `10` を入力します。

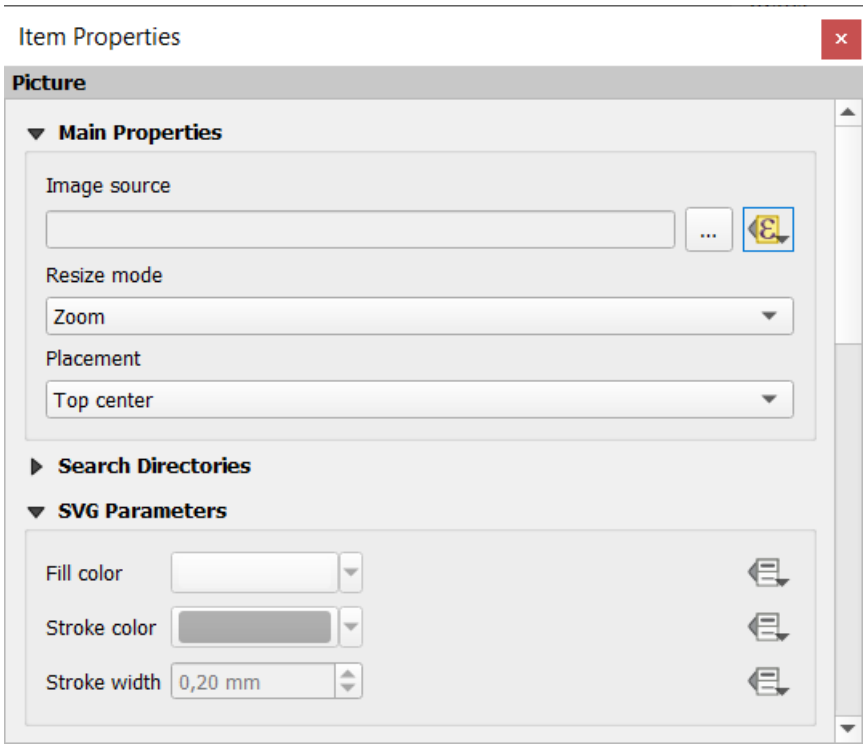
ページの中央にある地図の上にタイトルを追加してみましょう。ラベルアイテムを選択し、水平方向配置を **中央** に設定します。次にラベルを正しい位置に移動させ、**基準点** として中央のボタンを選択し、*X* フィールドに以下の式を指定します。

```
@layout_pagewidth / 2
```

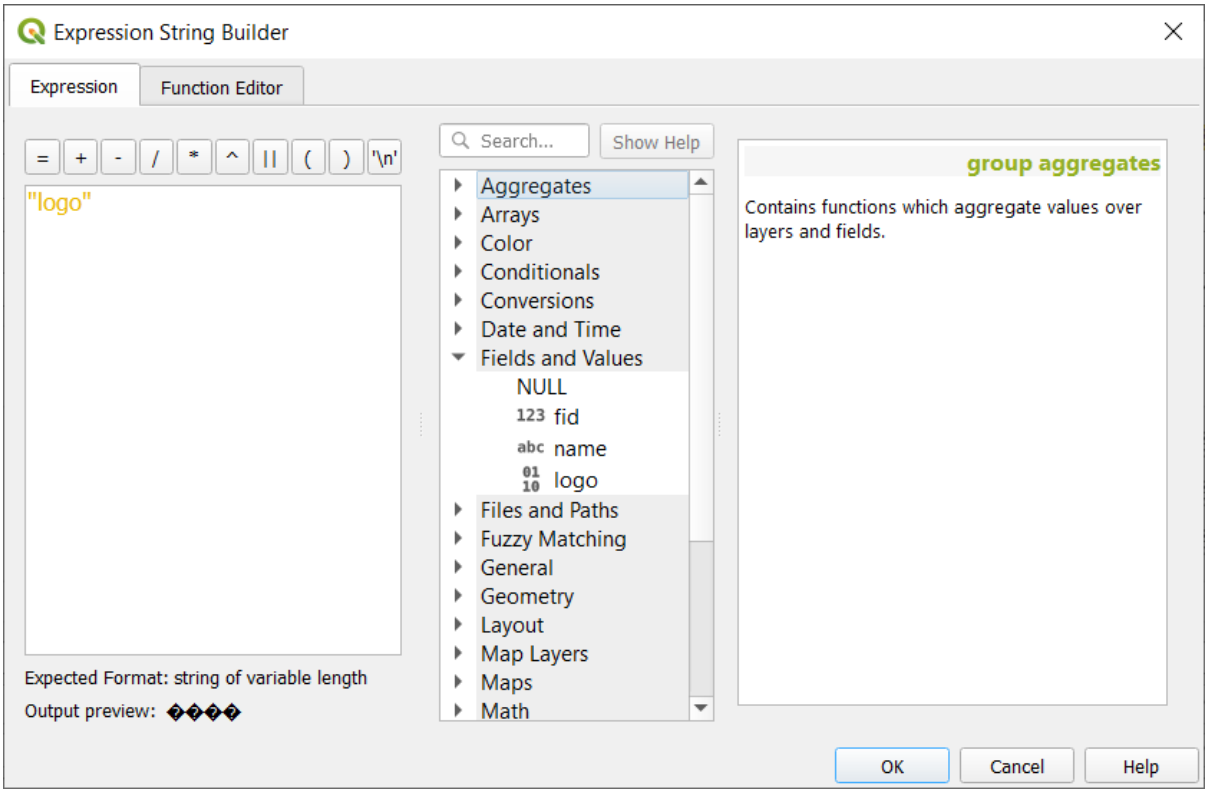
他のどんなレイアウトアイテムも同様の方法で位置を設定でき、これによって縦横どちらの向きでもレイアウトアイテムを正しく配置することができます。またさらに、タイトルを地物属性でカスタマイズしたり（使用例は **式を使ったラベルのカスタマイズ** を参照）、画像を変更したり、ページの向きに応じて凡例の列数を変更するなどの調整ができます。

ここで紹介した情報は、データによって定義された上書きオプションに関する素晴らしいブログ（英語・ポルトガル語） Multiple_format_map_series_using_QGIS_2.6 を更新したものです。

データによって定義された上書きボタンのもう一つの例として、画像の動的な変更への利用があります。以下の例では、フィールド型がバイナリの `logo` という名前の BLOB フィールドを含む GeoPackage レイヤを使用します（ **新しい *GeoPackage* レイヤを作成する** 参照）。各地物に対して別々の画像が定義されており、 **地図帳のプレビューと作成** で説明したように、地図帳でイテレーションできます。必要なことは、地図帳の設定を行った上で印刷レイアウトに画像アイテムを追加し、その **アイテムプロパティ** に移動することです。 **メインプロパティ** の **画像ソース** セクション内に、データによって定義された上書きボタンがあります。



次のウィンドウで **編集** を選択すると、 **式文字列ビルダ** が開きます。 **フィールドと値** セクションから、GeoPackage レイヤ内で定義された BLOB フィールドを見つけることができます。フィールド名の `logo` をダブルクリックし、 *OK* ボタンをクリックします。



地図帳は、この GeoPackage レイヤを **カバレッジレイヤ** として選択すると、BLOB フィールドのエントリを順番に取り出します（詳細については **地図帳のプレビューと作成** を参照）。

これらは、地図帳の高度な設定の使い方のほんの一例です。

地図帳のプレビューと作成

図 21.67: 地図帳ツールバー

地図帳の設定が完了し、レイアウトアイテム（地図、テーブル、画像など）と地図帳のリンクができたら、**地図帳**　**地図帳のプレビュー** を選択するか、または 地図帳のプレビュー アイコンをクリックすることで、全てのページのプレビューを作成することができます。その後、矢印ボタンを使って全ての地物を順番に確認することができます。

- 最初の地物
- 前の地物
- 次の地物
- 最後の地物

特定の地物を選択してプレビューするためにコンボボックスを使用することもできます。コンボボックスは、地図帳の **ページ名称** オプションで設定された式に基づいた地図帳地物の名前を表示します。

シンプルな印刷レイアウトと同様、地図帳の出力も様々な方法で生成できます（詳細については **出力の作成** を参照してください。ただし、ツールは **レイアウト** メニューのものを使う代わりに **地図帳** メニューまたはツールバーのものを使います）。

これにより、**地図帳**　**地図帳の印刷** を使用すれば、直接に地図帳を印刷できます。また、**地図帳**　*PDF* **として地図帳をエクスポート**... を使用すれば、PDF を作成することもできます。これを使うと、生成される PDF ファイル全てを保存するディレクトリを尋ねられます。ただし、 **可能であれば一つのファイルに出力** にチェックを入れている場合には、ファイル名を指定するように尋ねられます。

地図帳　**画像として地図帳をエクスポート**... や **地図帳**　*SVG* **として地図帳をエクスポート**... ツールの場合にも、フォルダを選択するよう尋ねられます。それぞれの地図帳地物の印刷レイアウトの全ページが、**地図帳** パネルで設定した画像ファイル形式または SVG 形式でエクスポートされます。

注釈: 複数ページの出力の場合には、印刷レイアウトの場合と同様に、地図帳は **一般設定** で説明される参照マップを含むページのみがワールドファイルを（各地図帳地物の出力に対して）得られます。

Tip: 特定の地図帳地物を印刷

地図帳地物の 1 つだけに対するレイアウトを印刷またはエクスポートしたい場合には、プレビューを開始し、ドロップダウンリストからエクスポートしたい地物を選択して、**レイアウト**　**印刷**（またはサポートしているファイル形式への **エクスポート**... ）をクリックするだけです。

地図帳の作成にプロジェクトで定義されたリレーションを使用する

HTML と Javascript の知識があるユーザーは、GeoJSON オブジェクトを操作し、QGIS プロジェクトで定義されたリレーションを使用することができます。この方法と、HTML に直接挿入された式を使用する方法との違いは、こちらの方が完全で非構造的な GeoJSON フィーチャーを扱える点です。つまり、既存の Javascript ライブラリや関数を使用して GeoJSON フィーチャー表現を操作できます。

以下のコードは、定義されたリレーションから関係するすべての子地物を含みます。JavaScript の `setFeature` 関数を使用すると、リレーションを好きな形式（リスト、テーブルなど）で表現する柔軟な HTML を作成することができます。このコードサンプルは、関係する子地物の動的な箇条書きリストを生成します。

```
// Declare the two HTML div elements we will use for the parent feature id
// and information about the children
<div id="parent"></div>
<div id="my_children"></div>

<script type="text/javascript">
   function setFeature(feature)
   {
     // Show the parent feature's identifier (using its "ID" field)
     document.getElementById('parent').innerHTML = feature.properties.ID;
     //clear the existing relation contents
     document.getElementById('my_children').innerHTML = '';
     feature.properties.my_relation.forEach(function(child_feature) {
     // for each related child feature, create a list element
     // with the feature's name (using its "NAME" field)
       var node = document.createElement("li");
       node.appendChild(document.createTextNode(child_feature.NAME));
       document.getElementById('my_children').appendChild(node);
     });
   }
</script>
```

地図帳の作成時には、親地物を含むカバレッジレイヤのイテレーションが行われます。各ページでは、親地物の識別子に応じて、関連する子地物の箇条書きリストが表示されます。

21.4 レポートの作成

このセクションでは、QGIS でのレポートの設定について説明します。

21.4.1 レポートとは？

定義から言えば、GIS レポートとはストーリー立てた情報を含む文書で、地図やテキスト、グラフィック、表などが含まれています。レポートはその場限りのものとして作成することもあれば、周期的、反復的、定期的に作成されたり、あるいは必要に応じて作成することもあります。レポートは、特定の期間やイベント、発生した出来事、対象、または場所などについて言及するものです。

QGIS においては、 **レポート** は **レイアウト** を拡張したものです。

レポート機能によって、ユーザーは GIS プロジェクトを簡単に、素早く、系統立った方法で出力することができます。

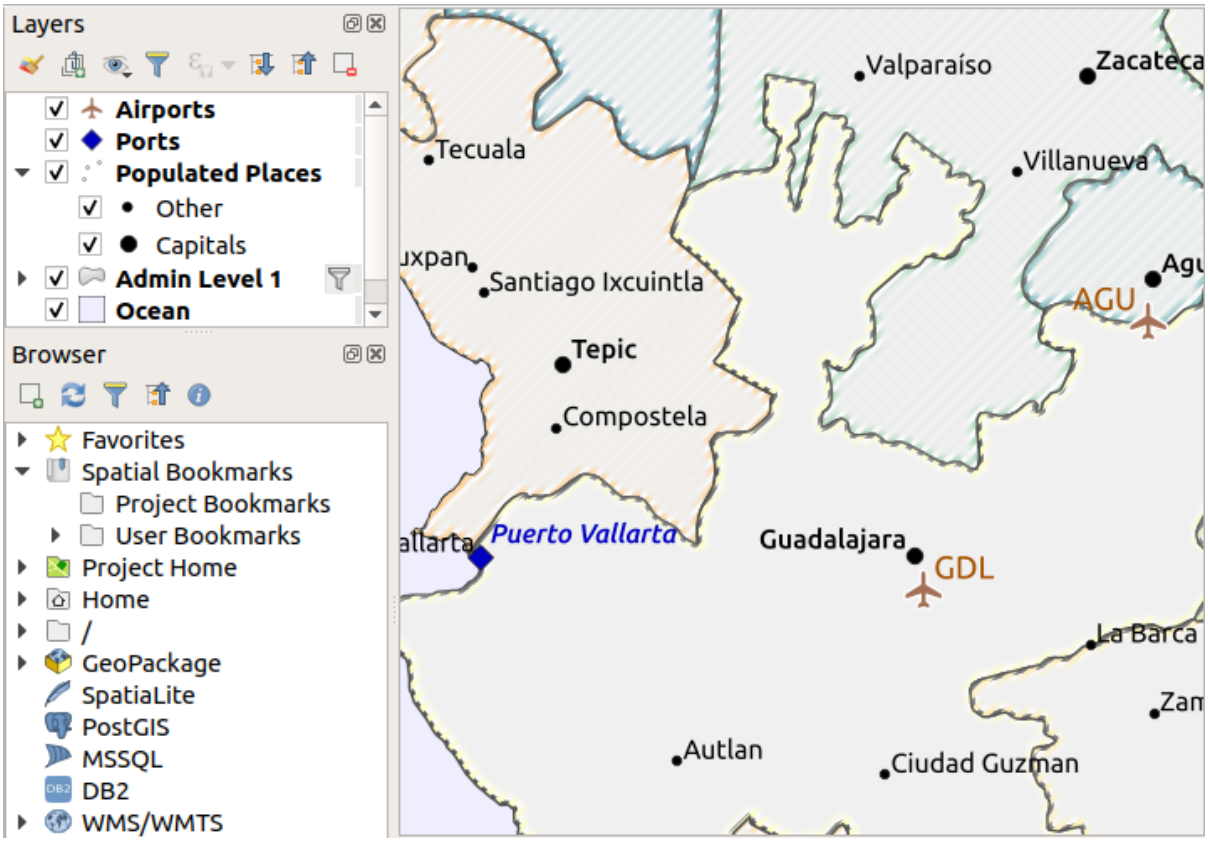
レポートは、 **プロジェクト** **新規レポート** または **プロジェクト** **レイアウトマネージャ** のダイアログ内から作成できます。

注釈: QGIS レポートの地図は、印刷レイアウトや地図帳の地図と同じように動作します。ここでは、QGIS レポートに固有の内容に焦点を絞って説明します。地図の操作方法については、 **印刷レイアウト** や **地図帳** のセクションを参照してください。

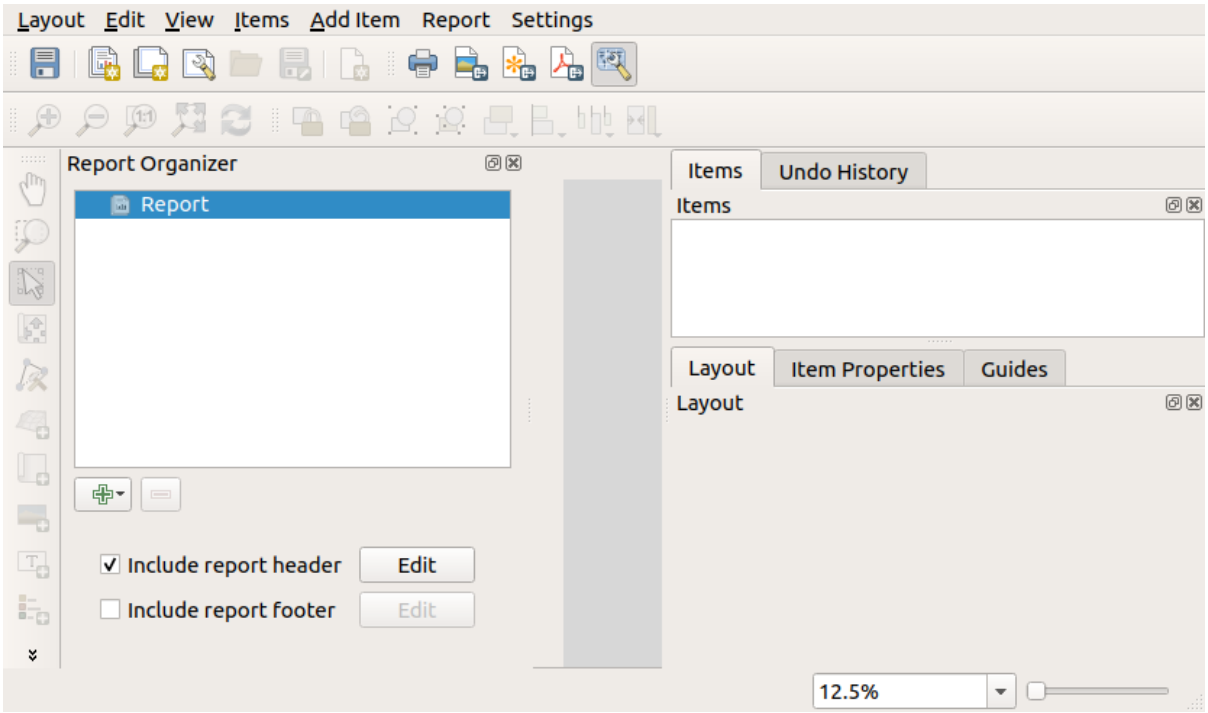
21.4.2 作成してみよう

レイアウトマネージャ ダイアログ内で、 **テンプレートから新規作成** から **空のレポート** のドロップダウンオプションを選択し、 **作成**... ボタンを押すことで、レポートを作成することができます。

この例では、Natural Earth dataset (1:10M) から、行政界、人口集中地域、港、空港などを使用しています。

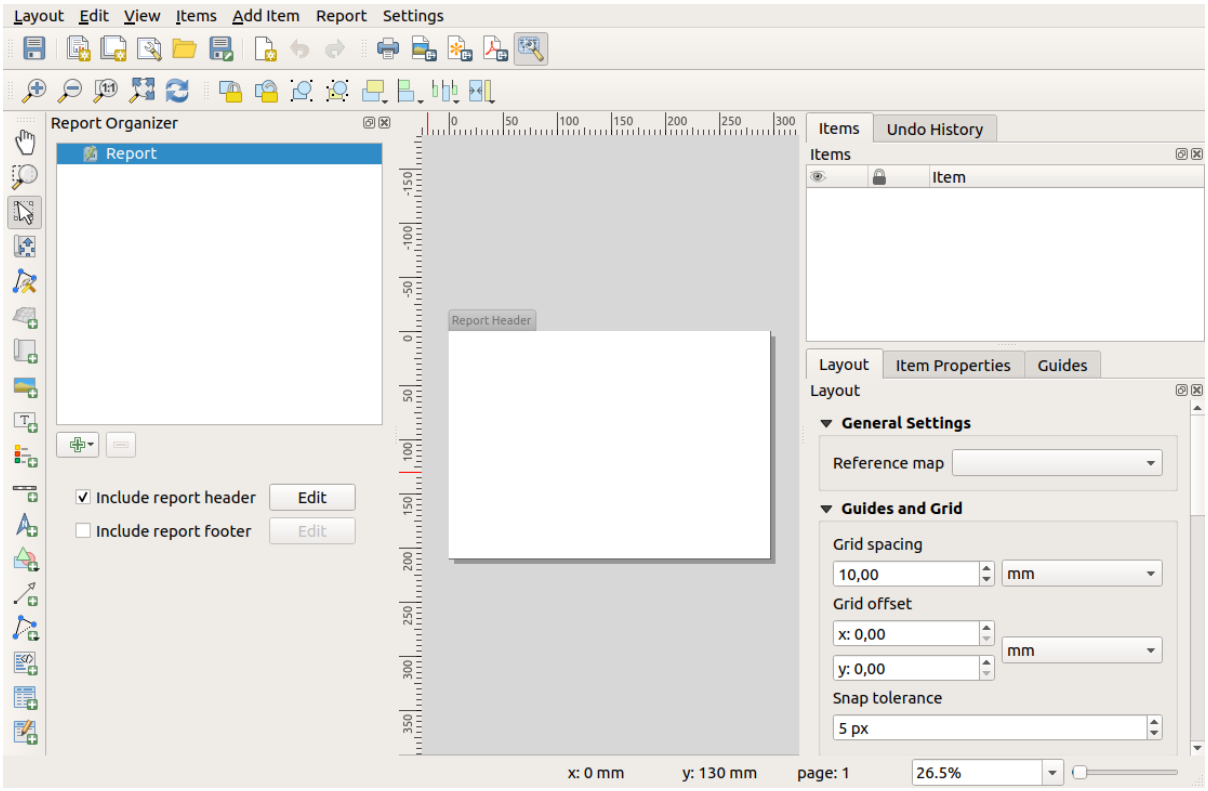


プロジェクト　**新規レポート** コマンドを使用すれば、空のレポートを作成できます。最初の状態では、見るべきものはあまりありません。表示されたダイアログは印刷レイアウトデザイナーとよく似ていますが、左側に **レポートオーガナイザ** パネルがある点が異なっています。



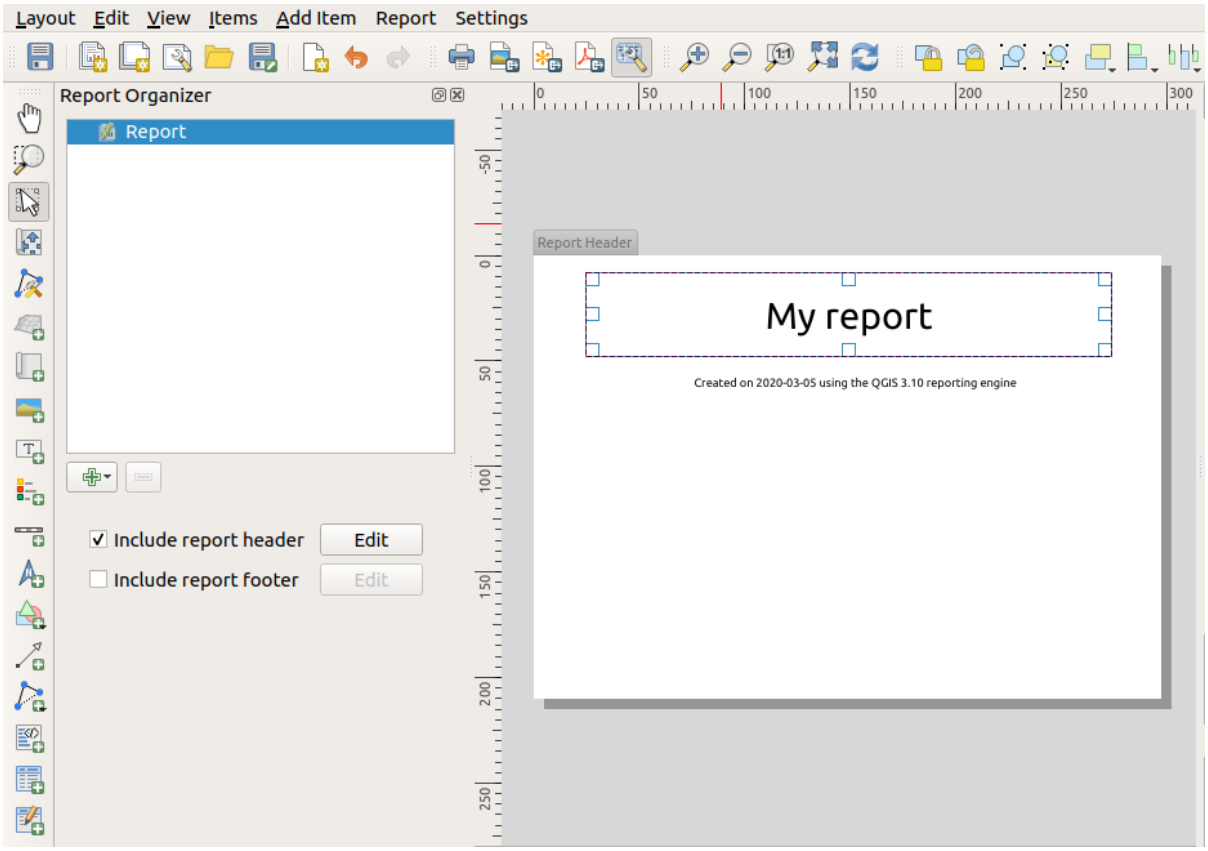
21.4.3 レポートワークスペースのレイアウト

QGIS レポートは、複数の入れ子になったセクションで構成されます。先ほど作成した空のレポートには、初期状態でメインレポートセクションのみがあります。このレポートセクションのオプションは、 **レポートヘッダを含む** と **レポートフッターを含む** のみです。これらのオプションを有効にすると、レポートの最初のページ（レポートの個々のパーツは必要に応じて複数ページにできます）にヘッダが含まれ、最後のページにはフッターが含まれます。ヘッダーを有効にして（ **レポートヘッダを含む** ）、その横にある **編集** ボタンを押してください。

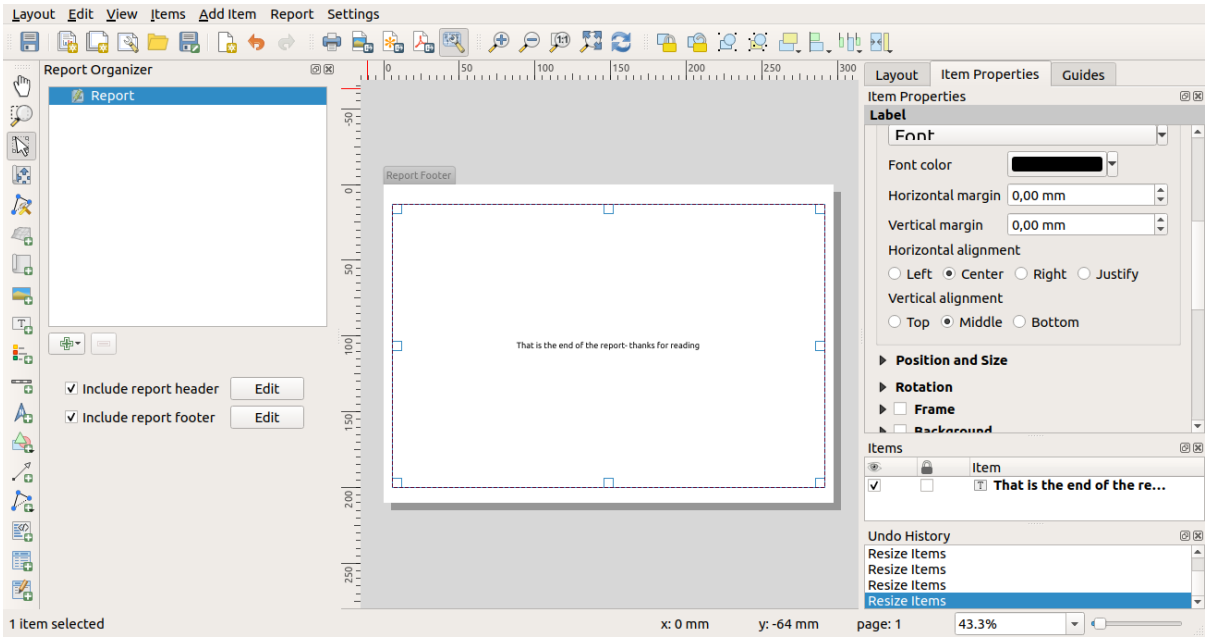


すると、いくつか変化が起こります。1 つは、 **レポートオーガナイザ** 内の **レポート** の横に編集中の鉛筆マークが表示され、このレポートセクションがデザイナーで編集中であることを示します。また、小さい **レポートヘッダ** のタイトルが付いた新しいページがあります。このページはデフォルトで **横** 向きですが、これはページ上を右クリックして **ページプロパティ** を選択すれば、このページの向き（やその他のプロパティ）の変更ができます。この操作によりページの **アイテムプロパティ** タブに移動し、ページの **大きさ** や **幅** 、 **高さ** などの指定ができます。

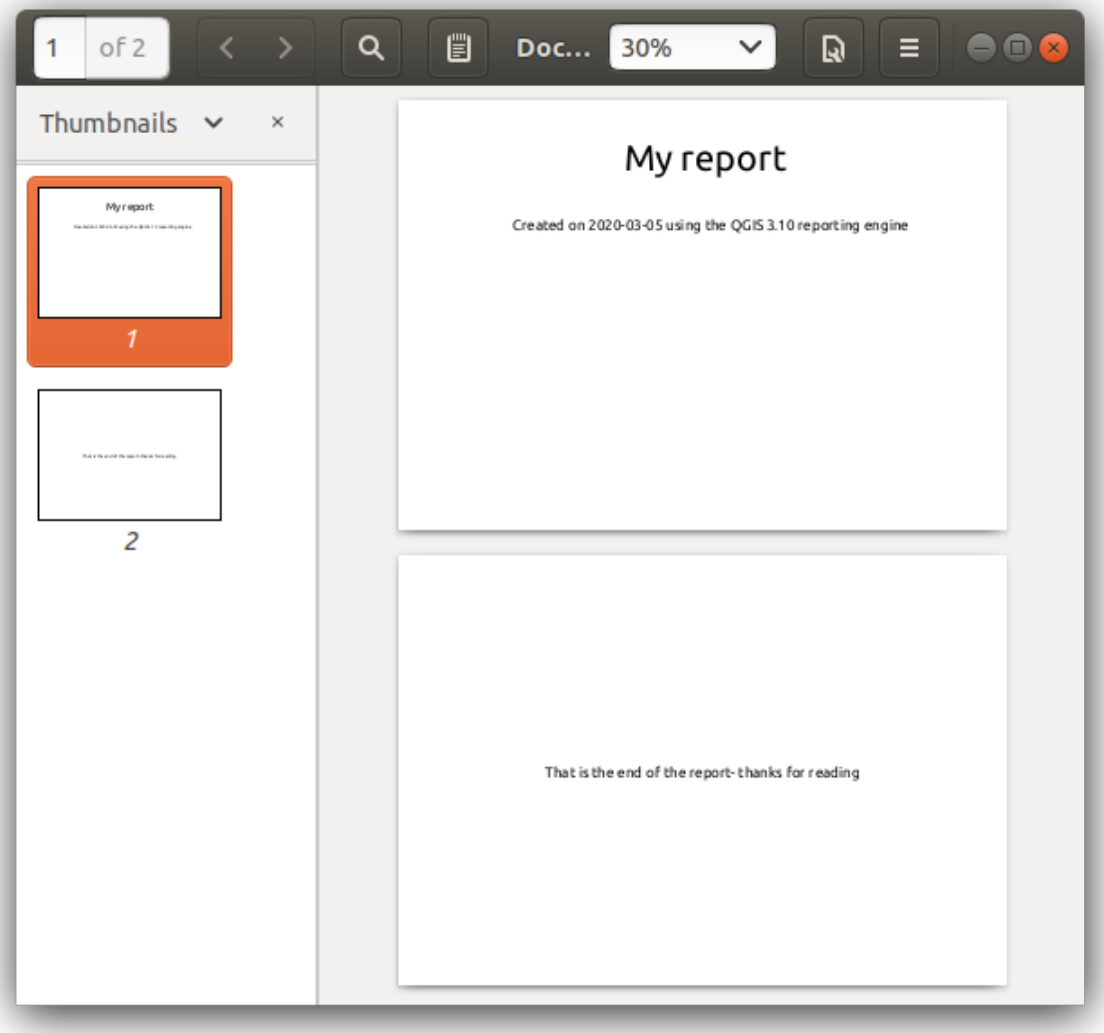
QGIS レポートでは、レポートの各コンポーネントは個々のレイアウトで構成されています。これらのコンポーネントは、標準的な印刷レイアウトと同じツールを使用して作成・変更ができます。従って、ラベルや画像、地図、テーブルなどを自由に組み合わせて使用できます。それでは、実際にレポートのヘッダにいくつかのアイテムを追加してみましょう。



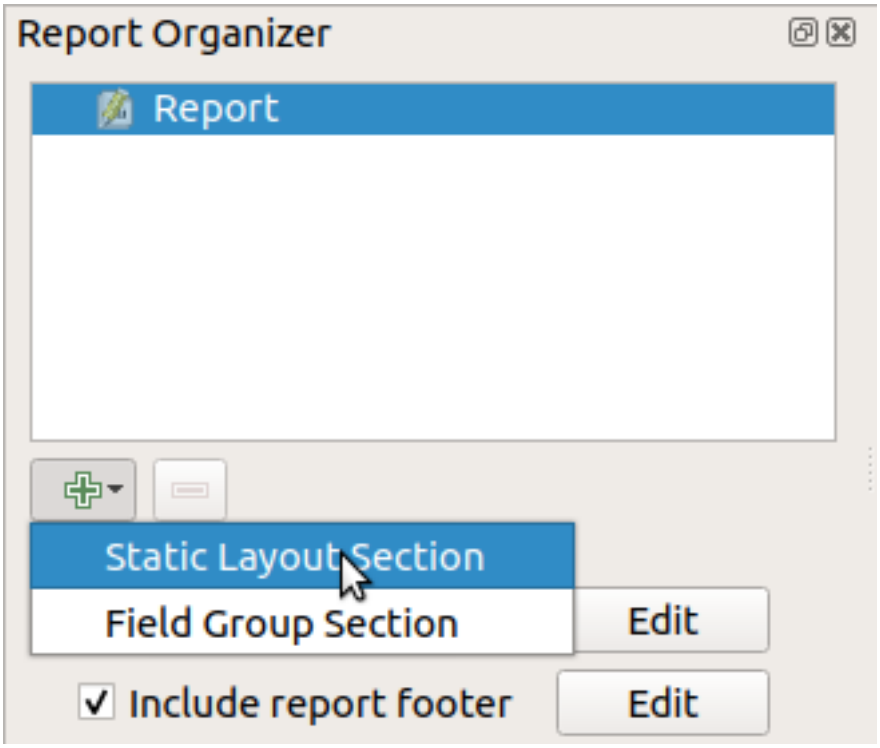
レポートにはシンプルなフッターも作成しましょう。 **レポートフッターを含む** オプションにチェックを入れ、 **編集** ボタンを押します。



先に進む前に、このレポートをエクスポートし、どんなものが得られるのか確認してみましょう。エクスポートは、 **レポート** メニューから行います。ここでは *PDF* **としてレポートをエクスポート**... を選択して、レポート全体を PDF ファイルへレンダリングします。結果は、ヘッダとフッターから構成された 2 ページの PDF という、あまり印象的ではないものです。



このレポートをもっと面白いものにしてみましょう。 **レポートオーガナイザ** セクションを追加 ボタンを押すと、レポートに追加する新しいセクションに関する選択肢が表示されます。

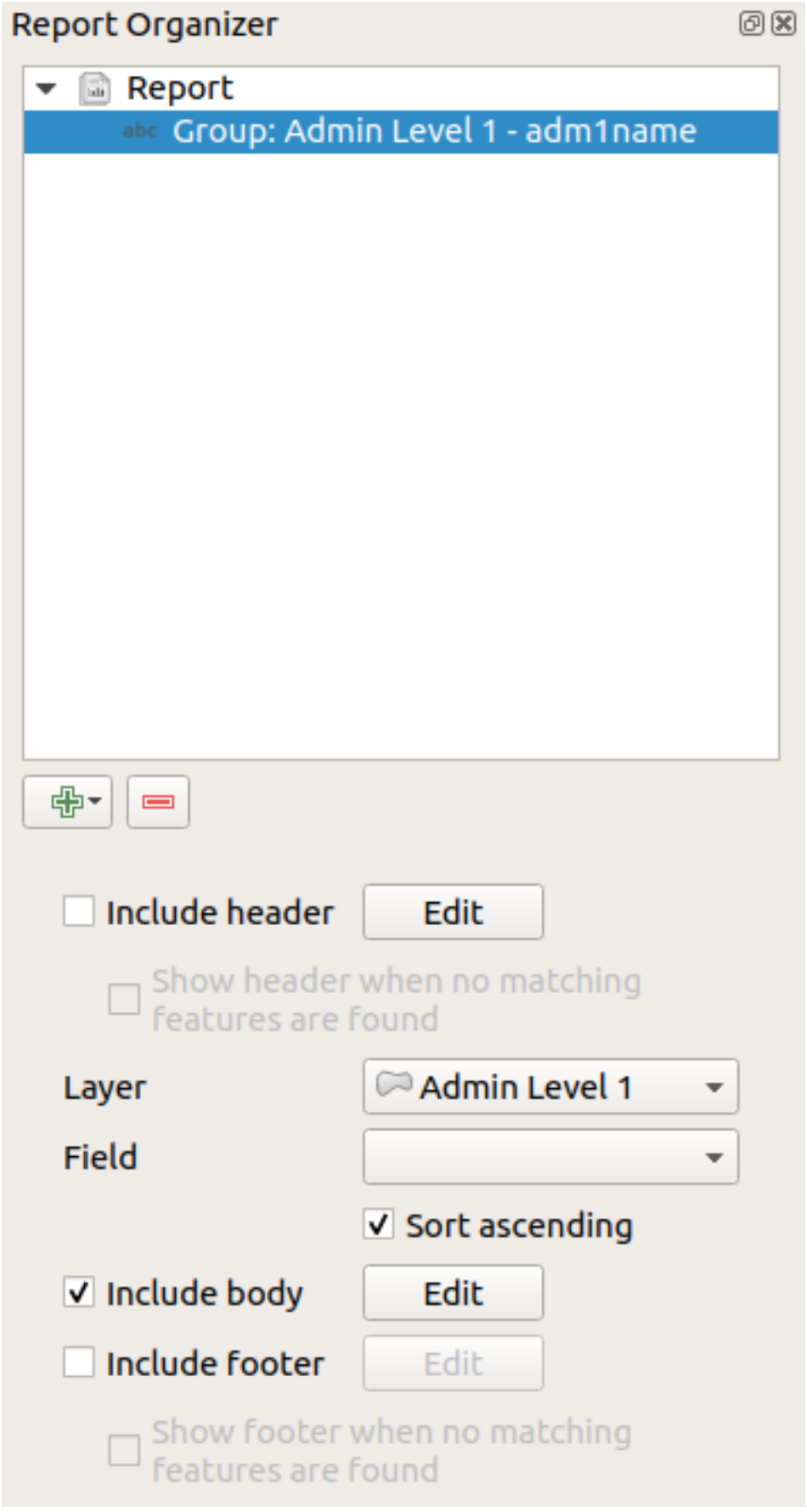


静的レイアウトセクション と **フィールドグループセクション** という 2 つの選択肢があります。

静的レイアウトセクション は、単一の静的なレイアウトボディを追加します。これは、レポートの途中で静的レイアウトを埋め込むために使用します。

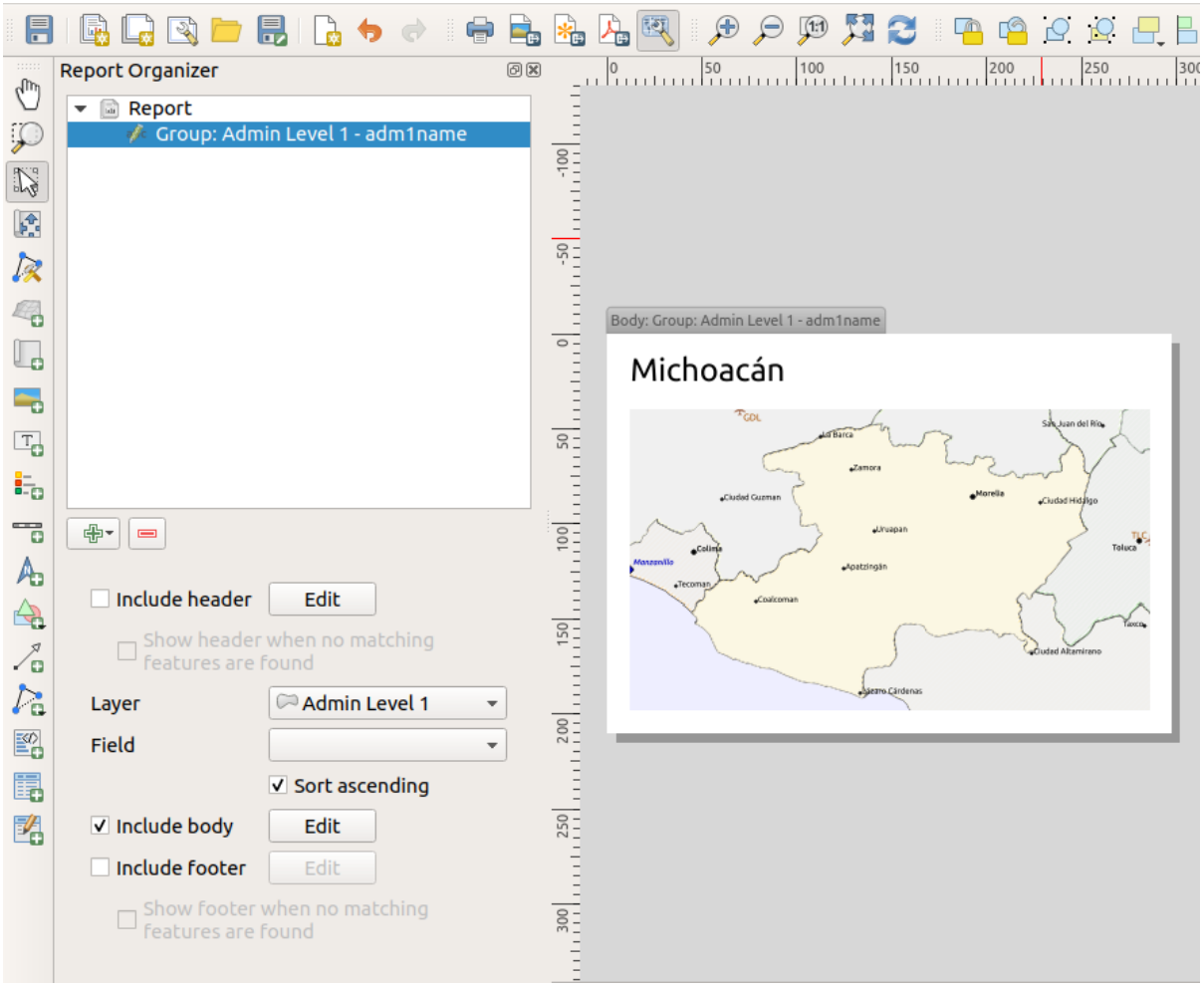
フィールドグループセクション は、レイヤの地物それぞれに対してボディレイアウトを繰り返します。地物は、 選択したグループ地物で並べ替えられます（昇順 / 降順ソートのオプション付き）。フィールドグループセクションが子セクション（例えば、異なるフィールドに対する別のフィールドグループセクション）を持つ場合、グループ地物に対してユニークな値を持つ地物のみが順番に取り出されます。これにより、ネストされたレポートを作成することができます。

ここでは、レポートに **フィールドグループセクション** を追加します。最も基本的なレベルでは、 **フィールドグループセクション** は **地図帳** と同等のものと考えることができます。順番に処理するためのレイヤを選択すると、レポートは見つかった地物ごとにセクションを挿入します。新しい **フィールドグループセクション** を選択すると、いくつかの新しい関連する設定が表示されます。



ここでは、*adm1name* フィールドの値を使用して「Admin Level 1」 レイヤの全ての州をイテレーションす

るようにフィールドグループを設定します。ヘッダ・フッターを含めるための同様のオプションもありますが、このセクションに **ボディ** を含めるための新しいオプションがあります。このオプションを使用して、ボディを編集しましょう。

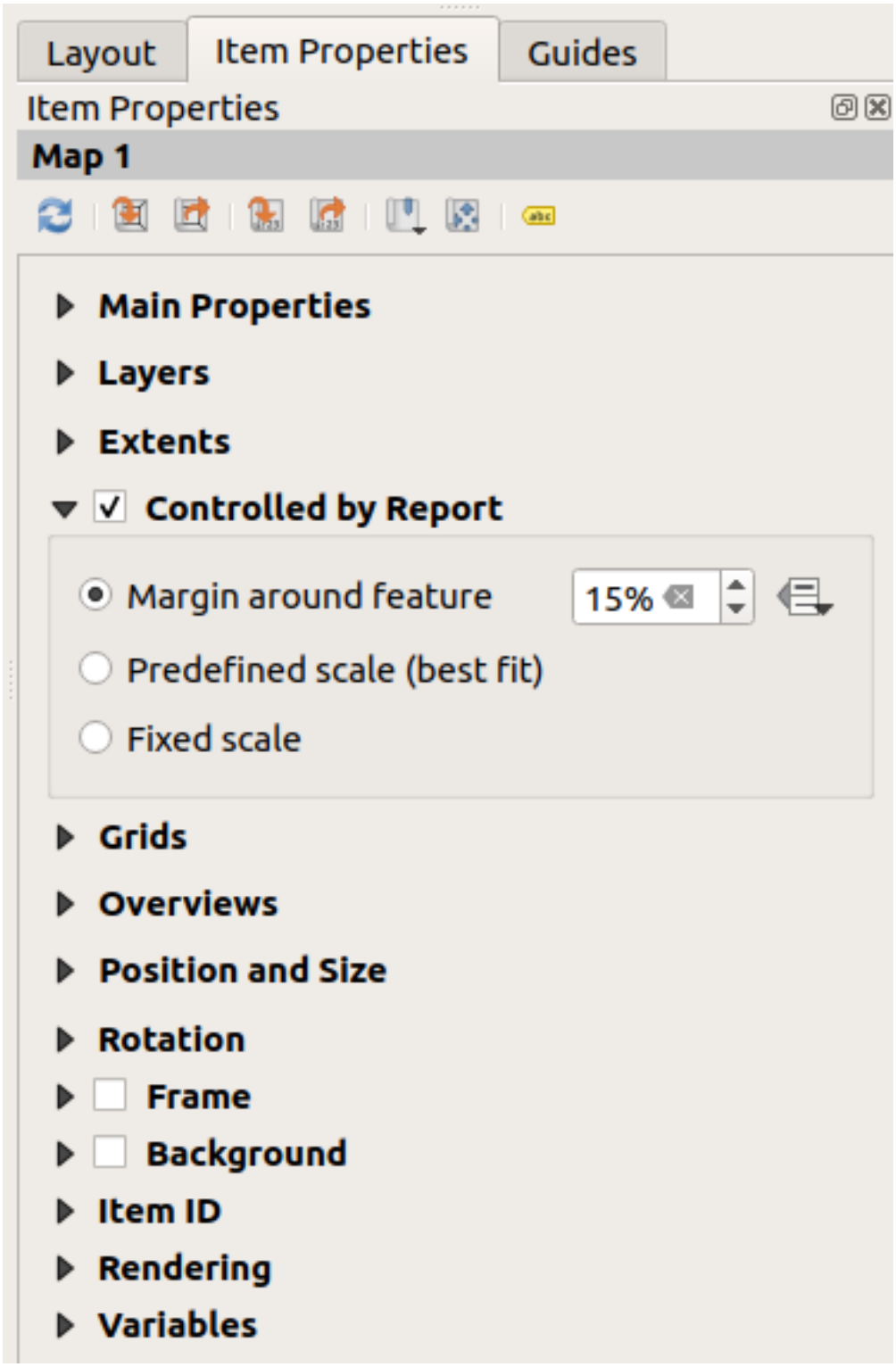


ボディは地図と州の名前を示すラベルで構成するものとしましょう。州の名前を入れるために、 **追加ラベルの追加** を選択し、 **メインプロパティ** で **式の挿入・編集**... を使ってデータ定義のテキストを追加します。

結果として次のような式が得られます（「name」は「Admin Level 1」レイヤの属性名で、州の名前を保持しています）。

```
[% "name" %]
```

地図は現在のレポート地物に従うように設定します（ **レポートによる制御** にチェックを入れて有効化します。 **地図帳による制御** にチェックが入っている場合に、地図帳の地図アイテムが現在の地図帳地物に従うのと同様です）。



ここまで設定できたところでレポートをエクスポートすると、以下のようなものが得られます。

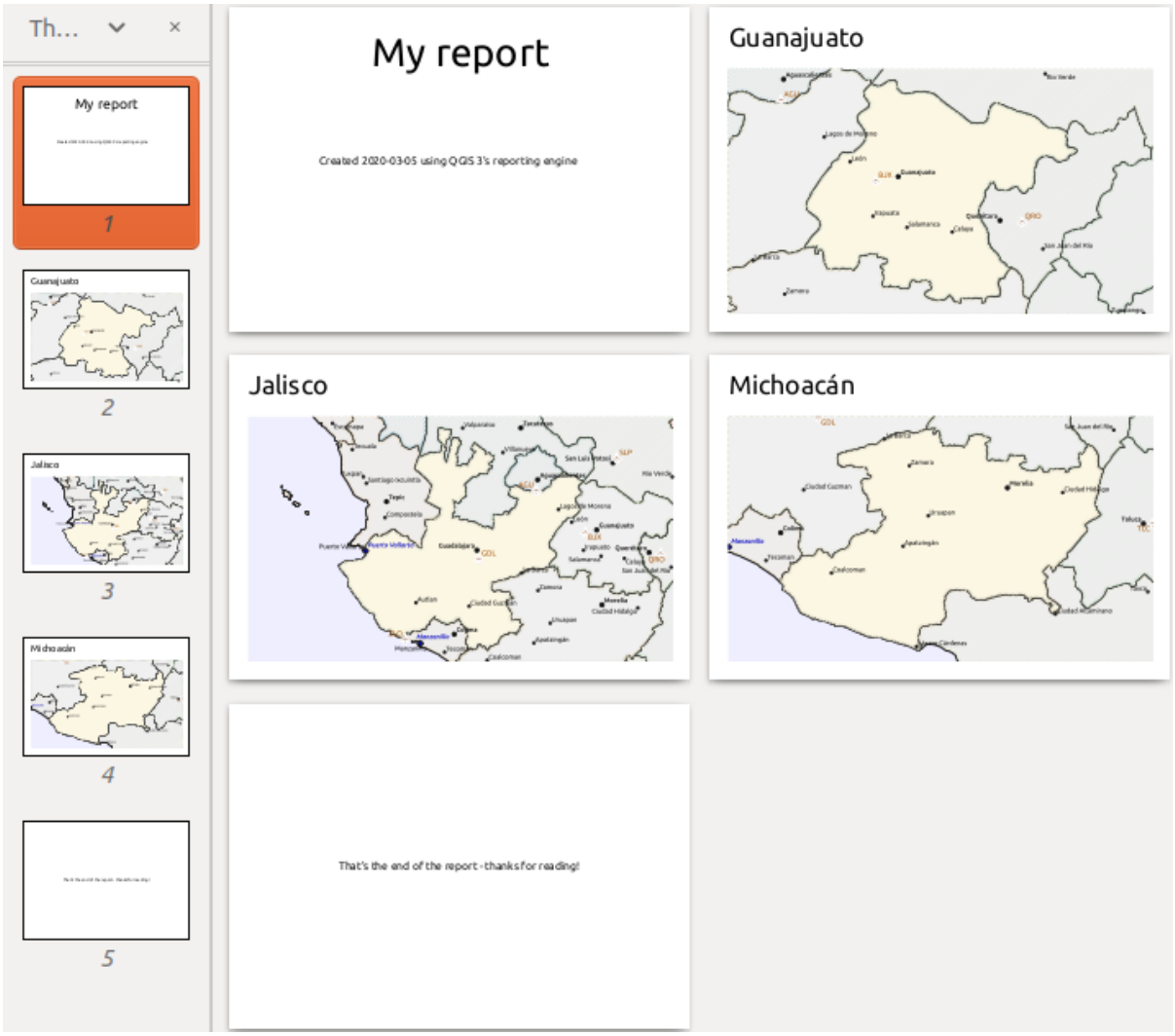
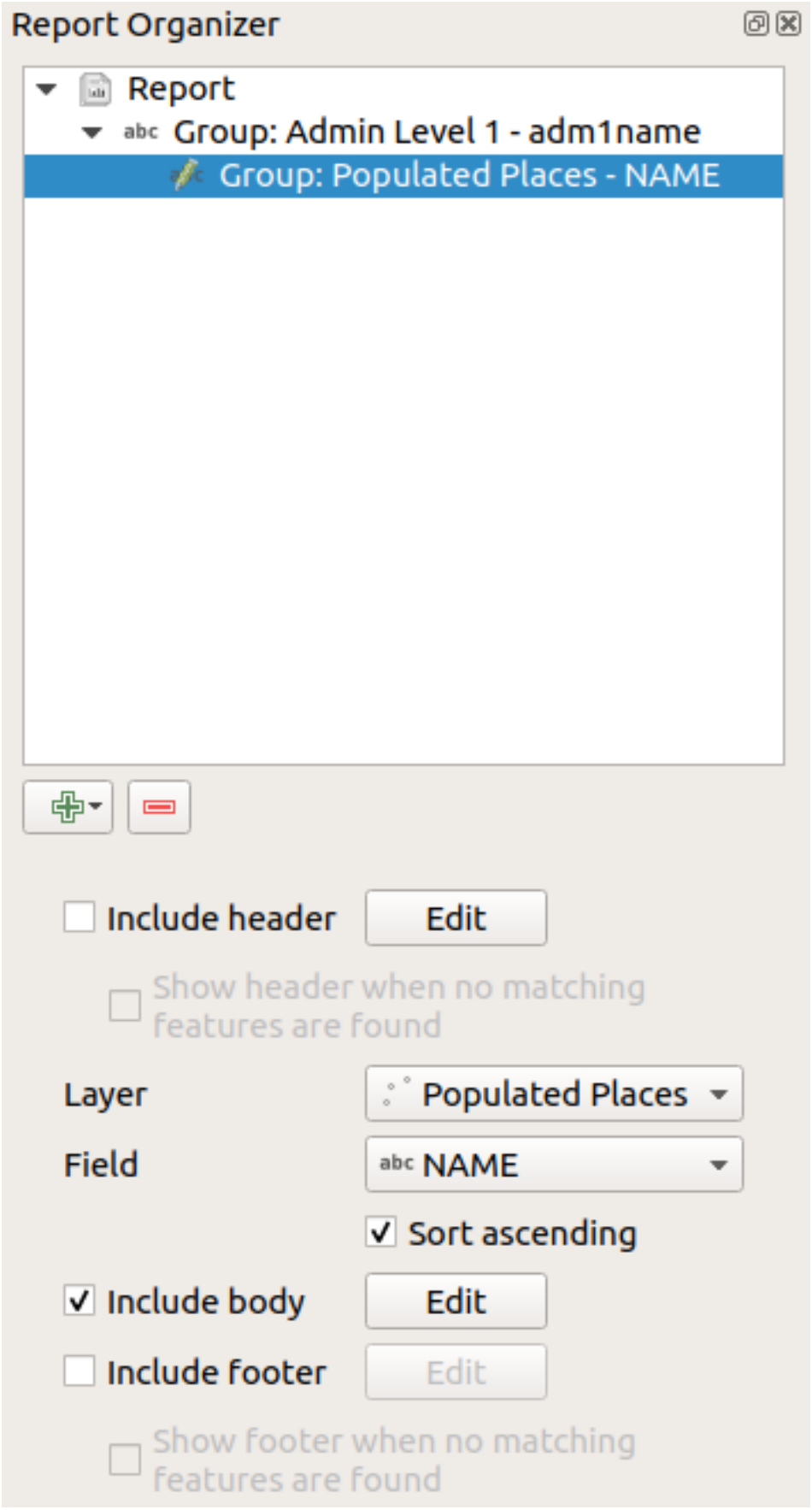


図 21.68: レポートのヘッダ、各州のページ、レポートのフッター

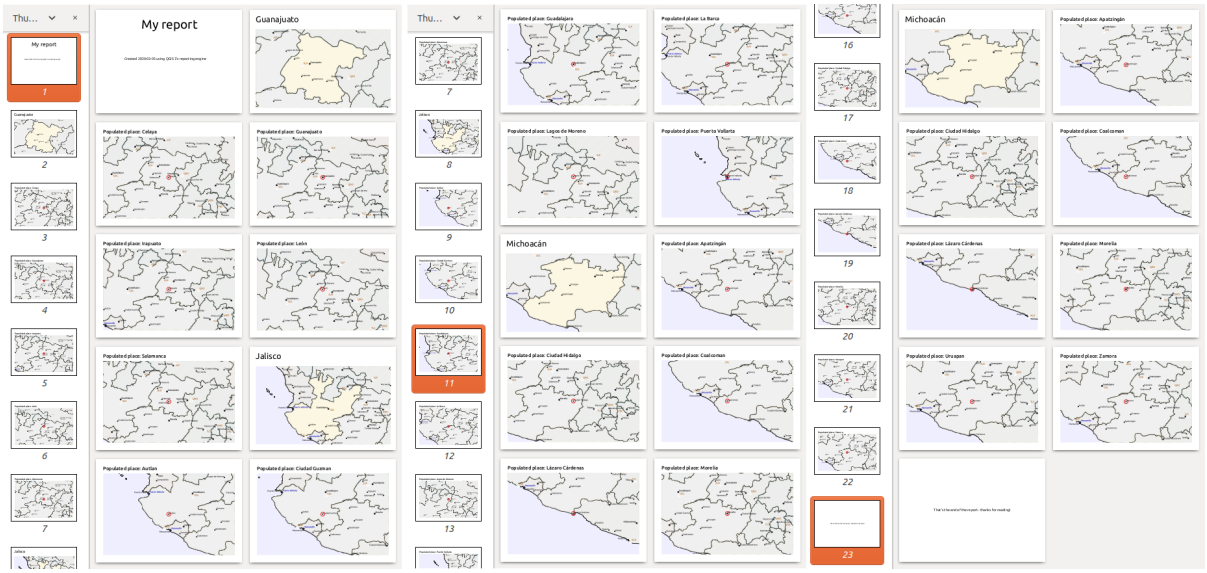
地図帳と似たようなものではありますが、ヘッダページとフッターページがあります。

州グループにサブセクションを追加して、レポートをより面白いものにしてみましょう。まず、レポートオーガナイザで「Admin Level 1」フィールドグループを選択し、次に セクションを追加 ボタンを押して、新しい **フィールドグループセクション** を追加します。



フィールドグループセクション の地物を反復処理する際には、その地物は親グループの定義フィールド（こ

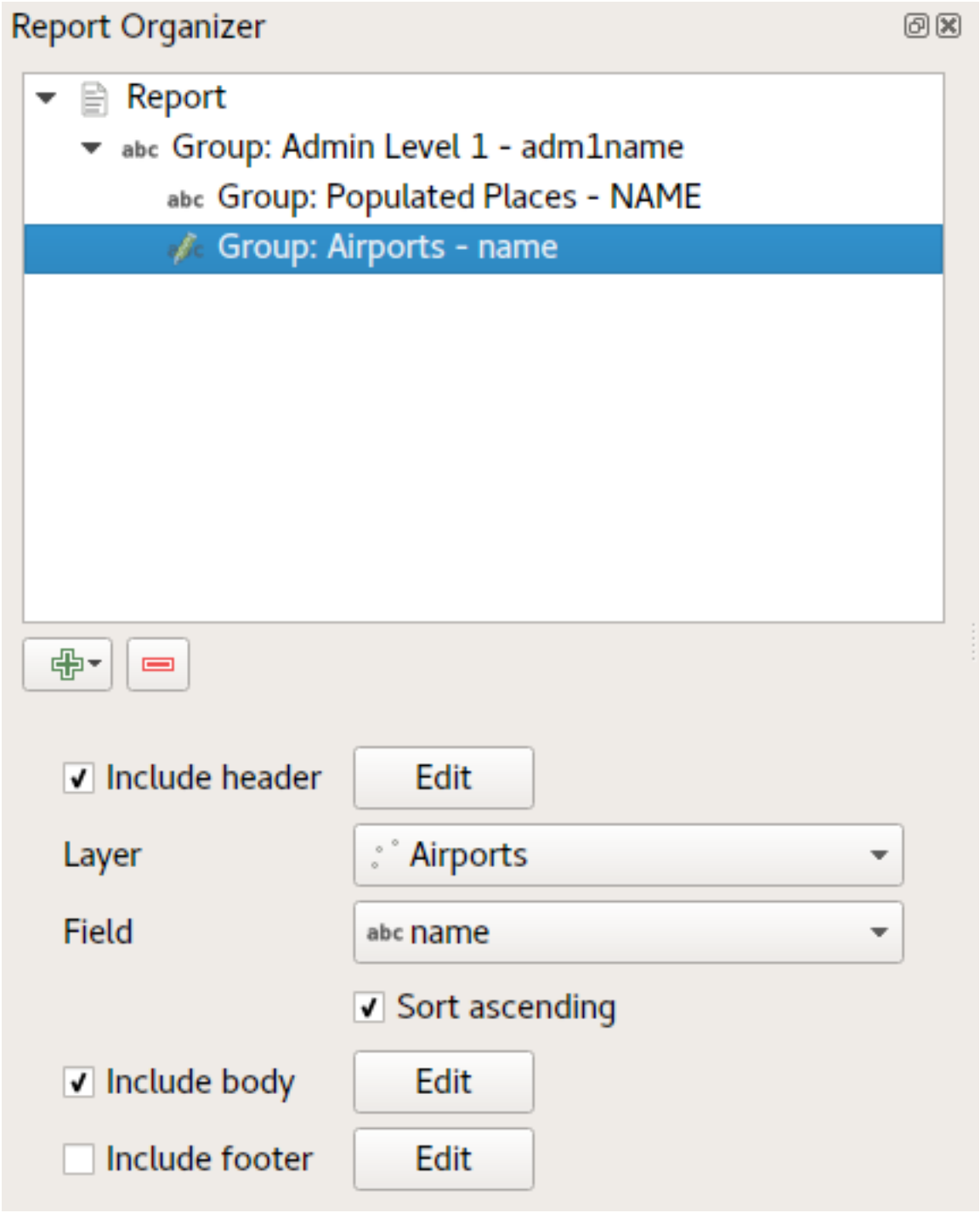
こでは `adm1name` ）にマッチするものでフィルタリングされます。この例では、追加したサブセクションが *Populated Places* レイヤを反復処理し、フィルタリング後に見つかった人口集中地区それぞれに対するボディセクションをレポートに組み込みます。ここでのマジックは、 *Populated Places* レイヤが親レイヤの定義フィールドと同じ名前の *adm1name* フィールドを持っており、各人口集中地区に対して、それが含まれている州名のタグとなっていることです（運が良ければ、データはすでにこのように構造化されていることでしょう。もし構造化されていない場合には、 **属性の空間結合** プロセシングアルゴリズムを実行して、独自のフィールドを作成してください）。このレポートをエクスポートすると、QGIS は最初に *Admin Level 1* レイヤから州を取得し、次に *adm1name* がマッチする *Populated Places* 全てに対して反復処理を行います。結果は以下のようになります。



ここでは Populated Places グループのために、この場所の地図と場所の属性に関するテーブルを含んだ、基本的なボディを作成しました。これで、レポートはヘッダ、最初の州のページ、続いて最初の州内の全ての人口集中地区のページ、それから他の州と人口集中地区のページで、最後にレポートのフッターという構成となりました。Populated Places グループにヘッダーを追加した場合には、以下に示す図のように、各州の人口集中地区が並ぶ直前のページに入ります。

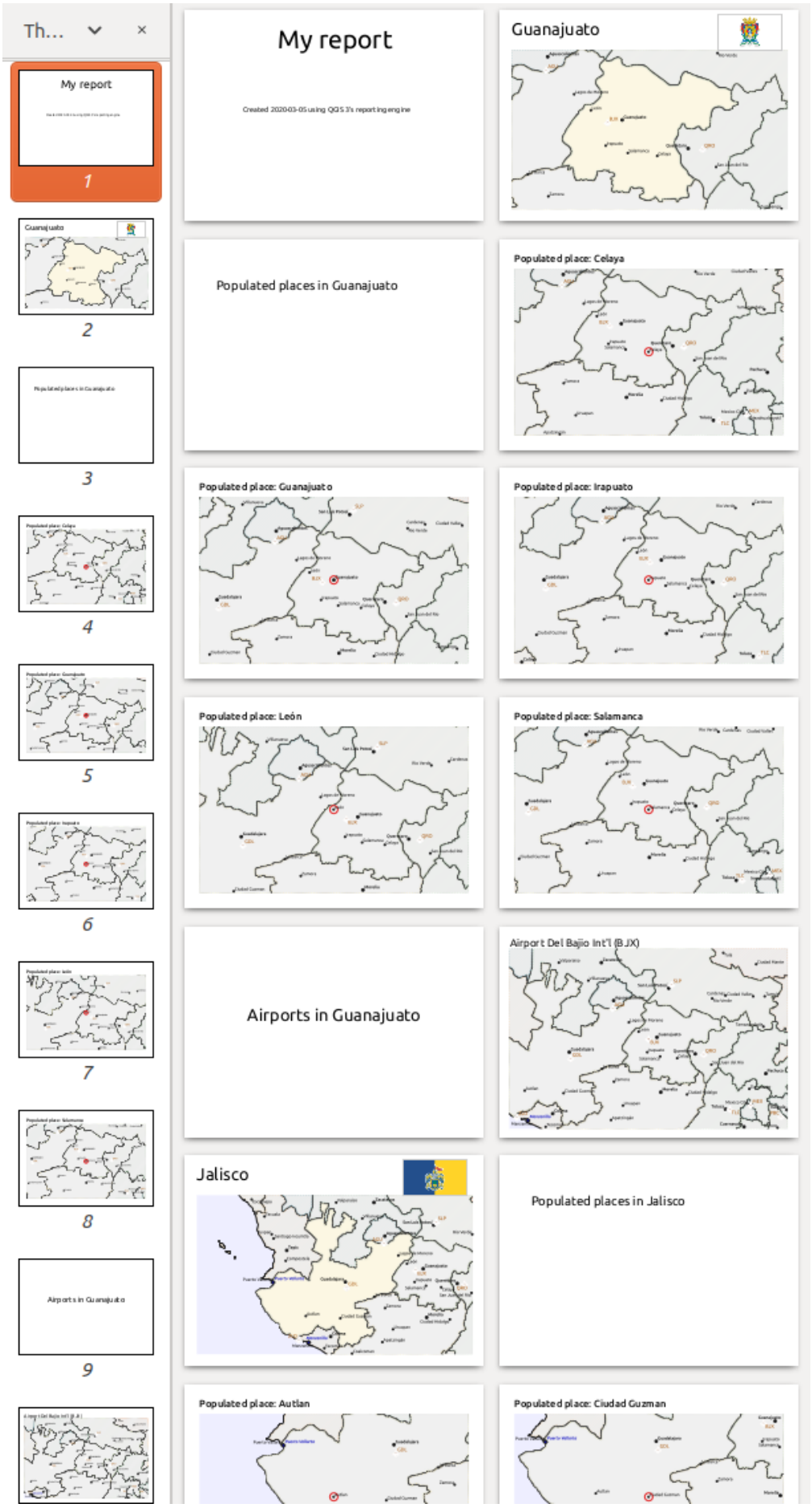
同様に、Populated Places グループに対するフッターは各州の最後の人口集中地区のページの後ろに挿入されます。

ネストされたサブセクションに加えて、レポート内のサブセクションは連続させることもできます。 *Admin Level 1* グループに対する 2 番目のサブセクションとして *Airports* を追加すると、（ *Airports* レイヤが親グループとリンクする *adm1name* 属性を持っているならば）レポートは最初に州の人口集中地区をリストし、続いてその州内の空港全てをリストして、それから次の州に移ります。



ここで重要なのは、*Airports* グループは *Admin Level 1* グループに対するサブセクションであり、*Populated Places* グループのサブセクションではないという点です。

これで、レポートは次のような構成になります（州旗も含まれていることに注意してください。このように地物別の画像を追加する手順については後述します）。


Th...
My report
1
2
3
4
5
6
7
8
9
My report
Created 2020-03-05 using QGIS 3's reporting engine
Guanajuato
Populated places in Guanajuato
Populated place: Celaya
Populated place: Guanajuato
Populated place: Irapuato
Populated place: León
Populated place: Salamanca
Airports in Guanajuato
Airport Del Bajio Int'l (BJX)
Jalisco
Populated places in Jalisco
Populated place: Autlan
Populated place: Ciudad Guzman

レポートに画像を含める

画像はレポートにおいて非常に有用です。QGIS では、レポートの静的な部分と動的な部分のどちらにも画像を使用できます。画像は標準的な印刷レイアウトと同じ方法で追加でき、レポートの静的な部分（および動的なレポート部分内の静的な画像）については、操作はそれだけです。

ただし、レポート地物に合わせて画像を設定したい場合には、含める画像を指定するために使用する属性をレイヤが持っている必要があります。

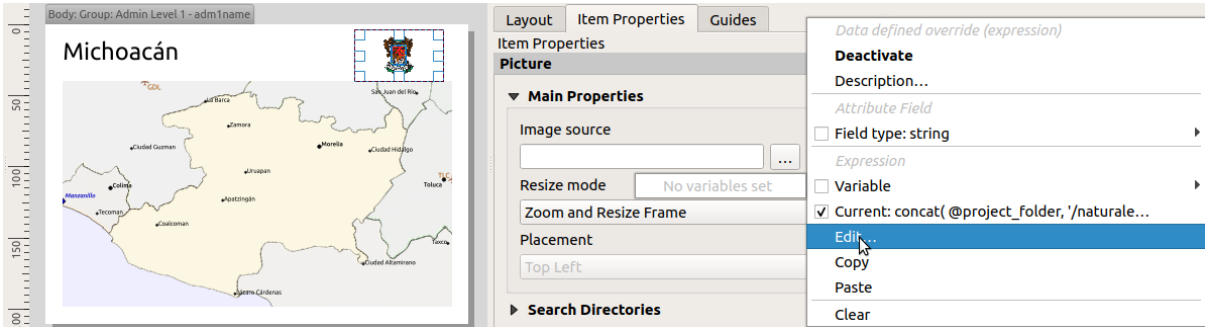
レポート内の画像については、QGIS では絶対ファイル名で指定します。

動的な画像を設定するには、まずは通常通り、グループのボディ部分に画像を追加します。画像の **アイテムプロパティ** において、 データによって定義された上書き ボタンを使用して **画像のソース** を設定し、画像の絶対パスを含む属性を選択するか、もしくは **編集**... を選択（して、画像の絶対パスを生成する式を入力）します。

以下は、文字列の連結を使用して画像の絶対パスを指定する式の例です。プロジェクトファイルが置かれているディレクトリ（ `@project_path` ）と、ファイル名を生成するための属性（ `adm1name` ）を使用しています（このケースでは、 `adm1name` 属性の文字列を大文字に変換し、 '_flag.png' を追加することでファイル名を生成しています）。

```
concat(@project_folder, '/naturalearth/pictures/' ,
       upper("adm1name"), '_flag.png')
```

これは、画像はプロジェクトファイルのあるディレクトリの `naturalearth/pictures` サブディレクトリ内にあることを意味しています。



地図内で現在のレポート地物を強調表示する

上で示したレポートでは、地図上でレポート地物をハイライト（州）と丸囲み（人口集中地区）を使って強調しています。地図上でレポート地物を（地図の中心に配置する以外の方法で）強調するためには、地図帳と同様に、地物の `@id` と `@atlas_featureid` の比較を用いてスタイルをデータで定義する必要があります。

例えば、レポート地物には他の地物よりも太いライン / 境界線を使用したい場合には、以下のように線の幅をデータ定義することができます。

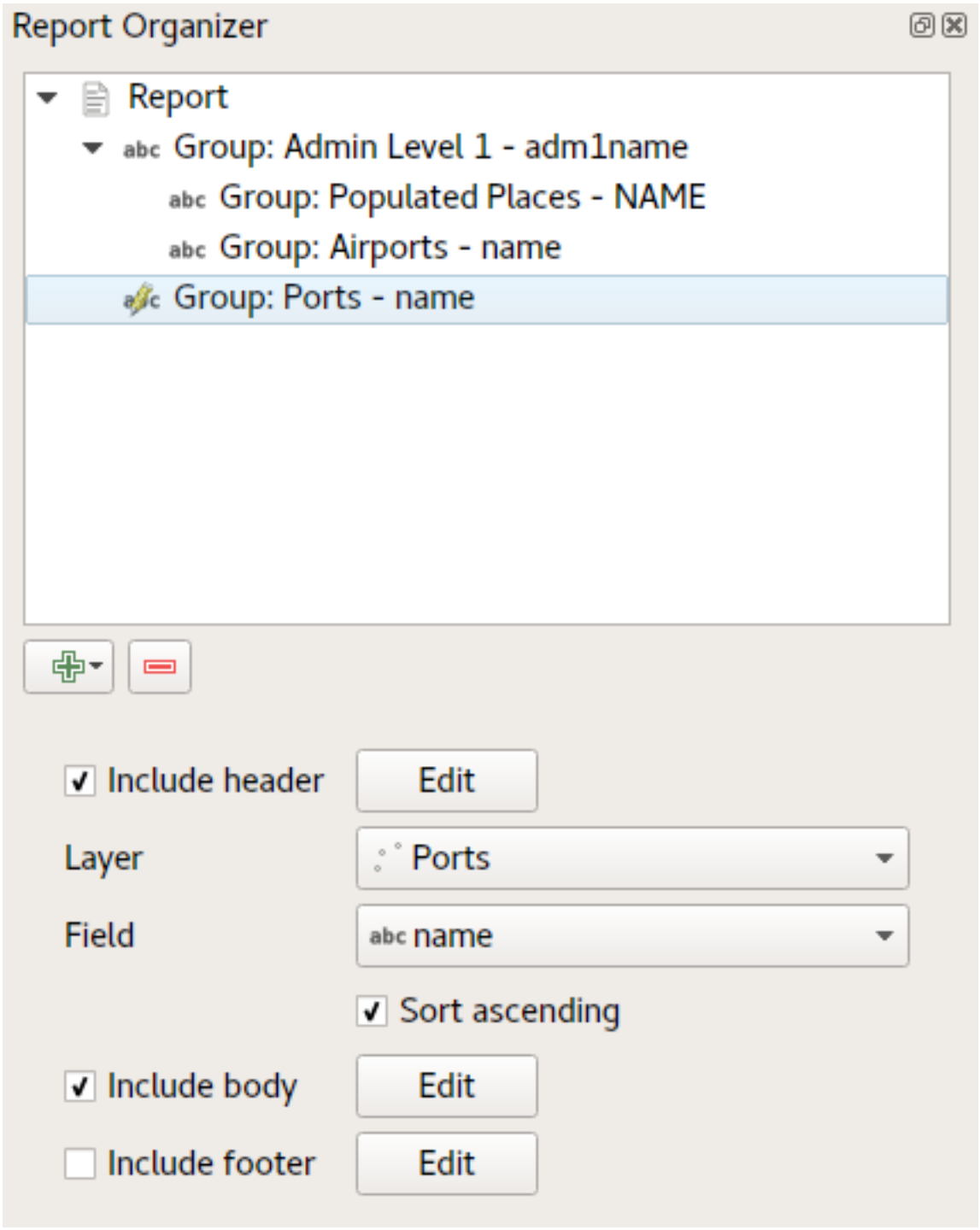
```
if($id=@atlas_featureid, 2.0, 0.1)
```

レポート地物は 2 単位のポリゴン境界線で表示され、その他の地物の線幅は 0.1 単位となります。色をデータ定義とすることもできます（レポート地物には不透明なダークマゼンタ、その他の地物には半透明の薄いグレーを使用するなど）。

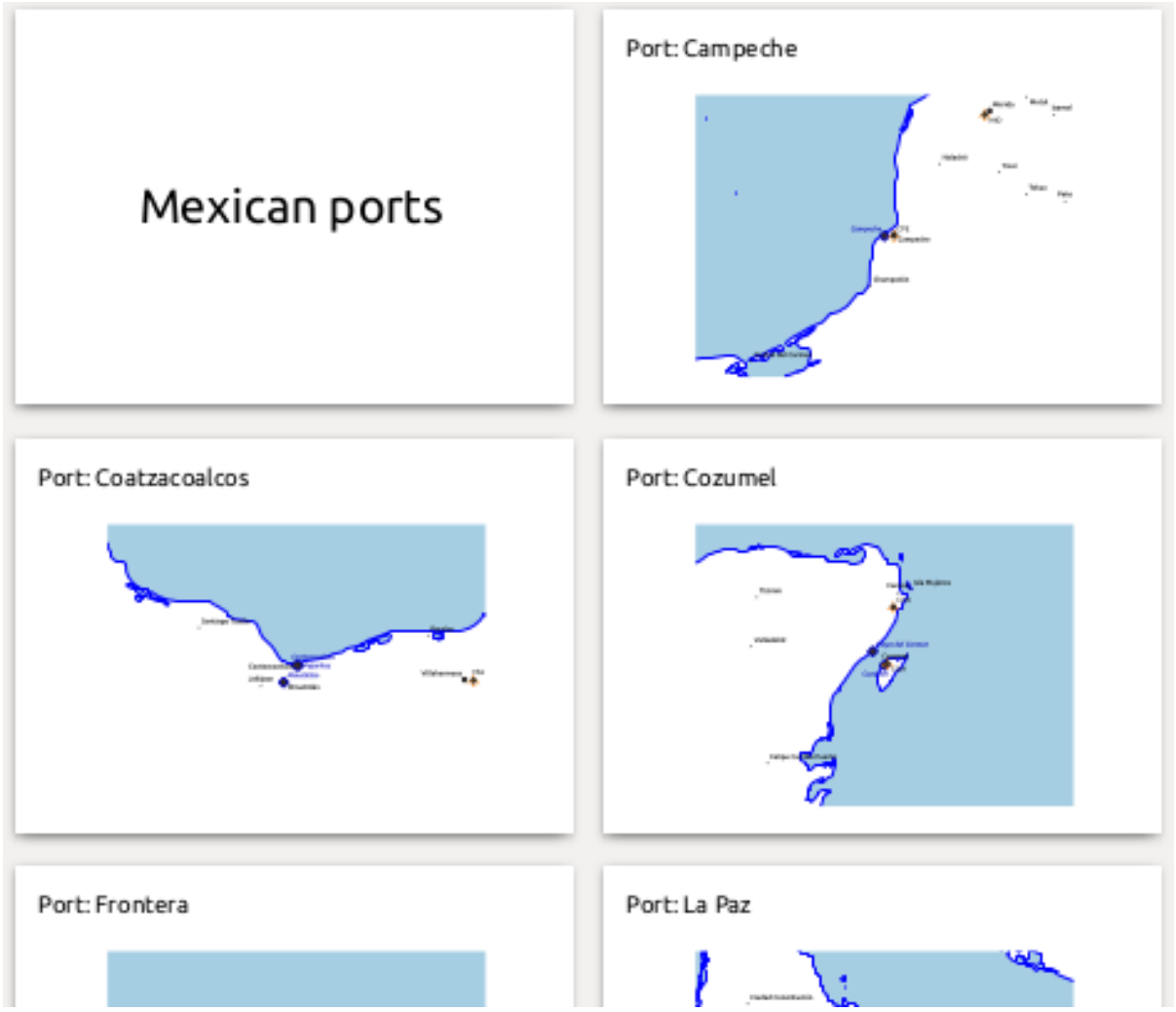
```
if($id=@atlas_featureid, '#FF880088', '#88CCCCCC')
```

最上位グループを追加する

ネストしたセクションや連続したセクションとセクションヘッダやフッターを組み合わせることで、レポートを非常に柔軟に作成できます。例えば、以下のレポートでは、メインレポートの子として *Ports* レイヤに関するフィールドグループをもう一つ追加しています。これで、州や人口集中地区、空港をリストアップした後、この地域のすべての港の概要リストを得ることができます。



このレポートの最後のパートは以下のようにエクスポートされます。



21.4.4 エクスポート設定

レポートをエクスポートする（ **レポート　画像 / *SVG* / *PDF* としてレポートをエクスポート**... ）時には、ファイル名の入力を求められた後、最適な出力が得られるようにエクスポート設定を調整することができます。

ここまででわかったように、QGIS のレポート機能は非常に強力で柔軟性に富んでいます！

注釈: これらの情報は、North Road のブログ Exploring Reports in QGIS 3.0 - the Ultimate Guide! から引用しています。

第 22 章　OGC / ISO プロトコルで作業する

Open Geospatial Consortium (OGC) は世界中の 300 以上の企業、政府、非営利団体、研究組織が集まった国際的組織です。ここのメンバーは地理空間コンテンツとサービス、GIS データの解析と交換のための標準の開発と実装を行っています。

GIS を含む、相互運用可能な位置・地理空間技術の特定のニーズに対応するため、地理的地物の基本的なデータモデルを記述する仕様が OGC によって次々に開発されました。詳細は https://www.ogc.org/ にあります。

QGIS でサポートされている重要な OGC 仕様は以下のとおりです:

- **WMS** --- Web 地図サービス (*WMS/WMTS* **クライアント**)
- **WMTS** --- Web 地図タイルサービス (*WMS/WMTS* **クライアント**)
- **WFS** --- Web 地物サービス (*WFS* **および** *WFS-T* **クライアント**)
- **WFS-T** --- Web 地物サービス - トランザクション的 (*WFS* **および** *WFS-T* **クライアント**)
- **WCS** --- Web カバレッジサービス (*WCS* **クライアント**)
- **WPS** ---ウェブ処理サービス
- **CSW** ---ウェブのためのカタログサービス
- **SFS** --- 単純地物 for SQL (*PostGIS* **レイヤ**)
- **GML** --- 地理情報記述言語

OGC サービスは、異なる GIS の実装とデータストアとの間の地理空間データを交換するためにますます使用されています。QGIS は（PostgreSQL/PostGIS のデータプロバイダーのサポートにより、*PostGIS* **レイヤ** のセクションを参照） **SFS** され、クライアントとして上記の仕様に対処できます。

QGIS サーバ 、UMN MapServer、または GeoServer がインストールされた Web サーバを使用すれば、WMS、WMTS、WFS、WFS-T、および WCS プロトコルを介して地図とデータを共有できます。

22.1 WMS/WMTS クライアント

22.1.1 WMS サポート概要

QGIS は現在、WMS 1.1、1.1.1 と 1.3 のサーバーを理解し WMS クライアントとして動作することができます。特に、そのような DEMIS として公にアクセス可能なサーバに対してテストされています。

WMS サーバーはクライアント（例えば、QGIS）の要求に応じて、指定された範囲、レイヤのセット、シンボル化スタイル、および透明度を持つ、ラスタ地図に作用します。WMS サーバーは、そのローカルデー

タソースを参照し、地図をラスタ化し、ラスタ形式でクライアントに送り返します。QGIS の場合、このフォーマットは一般的に JPEG または PNG になります。

WMS は一般的に、本格的な Web サービスではなく、REST（Representational State 転送）サービスです。そのため、QGIS が生成した URL を Web ブラウザで使うことで、QGIS が内部で使用するのと同じ画像を取得することができます。WMS サーバーにはいくつかのブランドがあり、それぞれが独自の WMS 標準の解釈を持っているので、トラブルシューティングするときにこれが役に立ちます。

WMS レイヤは、WMS サーバーにアクセスする URL を知っていればとても簡単に追加できます。サーバーに対してアクセスできる接続ができ、サーバーがデータ転送方式として HTTP を理解できれば大丈夫です。

さらに、QGIS は GetCapabilities リクエストがトリガーされない限り、WMS レスポンス（画像など）を 24 時間キャッシュします。GetCapabilities リクエストは、*WMS/WMTS* ダイアログの *Connect* ボタンを使用して WMS サーバーの能力を取得するたびにトリガーされます。これは、プロジェクトのロード時間を最適化するための自動機能です。プロジェクトが WMS レイヤーとともに保存されている場合、対応する WMS タイルは、24 時間以上経過していない限り、次にプロジェクトを開いたときにキャッシュから読み込まれます。

22.1.2 WMTS サポートの概要

QGIS は WMTS クライアントとしても動作します。WMTS は、地理空間データのタイルセットを配布するための OGC 標準です。WMTS ではタイルセットが事前に生成されており、クライアントはタイルの生成ではなく伝送を要求するだけなので、WMS より速く、効率的にデータを配信する方法です。一般的に WMS 要求はデータの生成と伝送の両方を含みます。タイル張りの地理空間データを閲覧する、非 OGC 標準のよく知られている例は、Google マップです。

ユーザーが望みそうなものに近い様々な縮尺でデータを表示するため、WMTS のタイルセットは、いくつかの異なる縮尺レベルで生成され、それらを要求する GIS クライアントのために利用できるようにされています。

この図はタイルセットの概念を示しています:

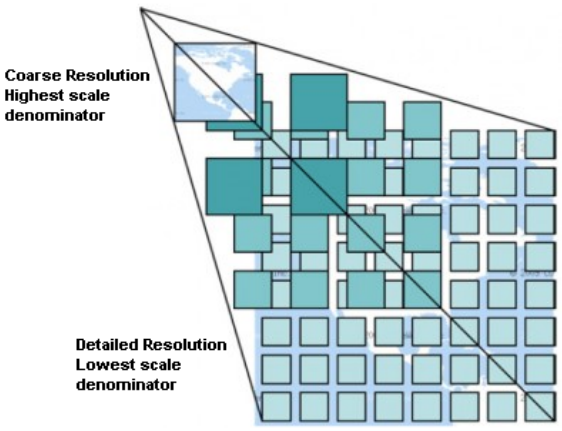


図 22.1: WMTS タイルセットの概念

QGIS がサポートしている 2 種類の WMTS のインタフェースは、キーと値のペア（KVP）と RESTful です。これら 2 つのインターフェイスは異なっており、それらは別々に QGIS に指定する必要があります。

1. **WMTS KVP** サービスにアクセスするには、QGIS ユーザーは WMS/WMTS インターフェイスを開き、WMTS タイルサービスの URL に次の文字列を追加する必要があります：

   ```
   "?SERVICE=WMTS&REQUEST=GetCapabilities"
   ```

   このタイプのアドレスの例は次のとおりです:

   ```
   https://opencache.statkart.no/gatekeeper/gk/gk.open_wmts?service=WMTS&
   ↪request=GetCapabilities
   ```

   この WMTS で topo2 レイヤをテストするためにうまく動作します。この文字列を追加することは WMS サービスの代わりに WMTS Web サービスを使うことを示します。

2. **RESTful WMTS** サービスは、異ったわかりやすい URL の形式を使います。この形式は OGC に推奨されています：

   ```
   {WMTSBaseURL}/1.0.0/WMTSCapabilities.xml
   ```

   この形式は、RESTful アドレスであることを認識するのに役立ちます。 RESTful WMTS は、フォームの URL フィールドの WMS セットアップでそのアドレスを追加するだけで QGIS でアクセスされます。オーストリアのベースマップを例にした、このタイプのアドレスは次のとおりです:

   ```
   https://maps.wien.gv.at/basemap/1.0.0/WMTSCapabilities.xml
   ```

注釈: WMS-C と呼ばれるいくつかの古いサービスもまだ見つけることができます。これらのサービスは WMTS と非常によく似ています（つまり、目的は同じですが、動作が少し異なります）。 これらは WMTS サービスと同じように管理できます。 URL の最後に「? tiled = true」を追加するだけです。この仕様の詳細については、 https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification 参照してください。

WMTS を読んだときに, WMS-C についても考えるでしょう.

22.1.3 WMS/WMTS サーバーを選択する

QGIS で WMS/WMTS 機能を初めて使用するとき、定義されたサーバーはありません。

ですから目標のサーバーへの接続を作る必要があります:

1. 次のいずれかの方法で **データソースマネージャ** ダイアログの *WMS/WMTS* タブに移動します:
   - データソースマネージャを開く ボタンをクリック（または Ctrl+L を押）し、タブを有効化する
   - **レイヤ管理** ツールバーの WMS レイヤを追加する ボタンをクリックする
   - **レイヤ** **レイヤを追加** *WMS/WMTS* **を追加**... メニューを選択する
2. **レイヤ** タブから **新規** を押します。 **新規** *WMS/* **接続を作成** ダイアログが表示されます。

Tip: **ブラウザパネル** 内の *WMS/WMTS* エントリーを右クリックして、 **新規接続...** を選択すると、 **新規** *WMS/* **接続を作成** ダイアログが表示されます。

3. 次に、下にあるように、接続したい WMS サーバーに接続するためにパラメータを入力します:

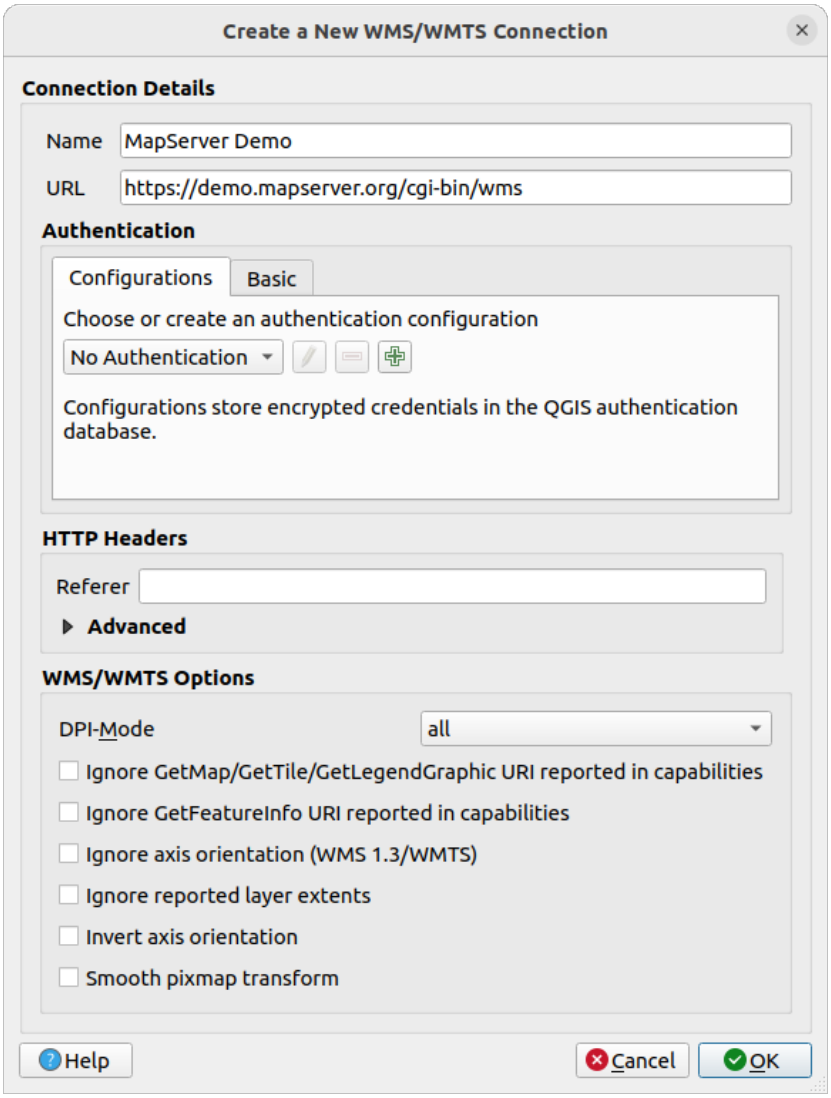


図 22.2: WMS サーバーへの接続を作る

- **名前**: 接続の名前。この名前は WMS サーバーを見分けられるよう、サーバー接続ドロップダウンボックスで使われます。
- *URL*: データを提供するサーバーの URL。これは解決可能なホスト名でなければなりません。telnet 接続を開いたりホストに ping するときに使うものと同じように、ベース URL だけの形式です。たとえば `request=GetCapabilities` や `version=1.0.0` のようなフラグメントが URL にあってはいけません。
- **認証**（オプション）: *stored configuration* または **ユーザー名** と **パスワード** によるベーシック認証を使います。

> **警告:** **認証** タブで **ユーザー名** と **パスワード** を入力すると、保護されていない資格情報が接続構成に保持されます。たとえば、プロジェクトファイルを誰かと共有した場合、これらの **認証情報が見えてしまいます** 。したがって、代わりに **認証設定** （ **設定** タブ）に資格情報を保存することをお勧めします。詳細は **認証システム** 参照。

- HTTP **リファラー**
- *DPI-Mode*: 有効なオプションは、 **すべて**, **オフ**, **QGIS**, **UMN** および **GeoServer** です
- ☐ *capabilities* **の** *GetMap/GetTile/GetLegendGraphic URI* **を無視**: チェックした場合、上の *URL* フィールドから URI を使います。
- ☐ *capabilities* **で返答された** *GetFeatureInfoURI* **を無視する** ：チェックした場合、上記の *URL* フィールドから指定された URI を使用します。
- ☐ **軸方位を無視する** *(WMS 1.3/WMTS)*
- ☐ **レイヤ範囲を無視**: ラスタレイヤによって報告される範囲はレンダリングできる実際の領域よりも小さい場合があるため（特に、データ範囲よりも多くのスペースを必要とするシンボルを使用する WMS サーバーの場合）、このオプションをオンにすると、報告された範囲にラスタレイヤがトリミングされず、これらのレイヤの境界で地図記号が切り捨てられるのを避けられます。
- ☐ **軸方位を逆にする**
- ☐ *Smooth pixmap transformation*

4. *OK* を押します。

新しい WMS/WMTS サーバー接続が作成されると、将来の QGIS セッションのために保存されます。

インターネットから WMS サービスを受信できるようにプロキシサーバーを設定する必要がある場合は、オプションでプロキシサーバーを追加できます。 **設定** **オプション** をクリックして **ネットワーク** タブを選択してください。そこでは、 ☑ **ウェブアクセスにプロキシを使用** 設定することで、プロキシ設定を追加しそれらを有効にできます。 **プロキシタイプ** ドロップダウンメニューから正しいプロキシタイプを選択していることを確認してください。

22.1.4 WMS/WMTS レイヤを読み込む

パラメータを正しく入力したら、 **接続** ボタンを使用して、選択したサーバの機能を取得できます。これには、画像のエンコーディング、レイヤ、レイヤスタイル、投影法があります。これはネットワーク操作であるため、応答の速度は WMS サーバへのネットワーク接続の品質によって異なります。 WMS サーバからデータをダウンロードしている間、ダウンロードの進行状況はメインの QGIS ダイアログの左下隅に表示されます。

あなたの画面は 図 22.3 のように、WMS サーバーからの応答を表示するはずです。

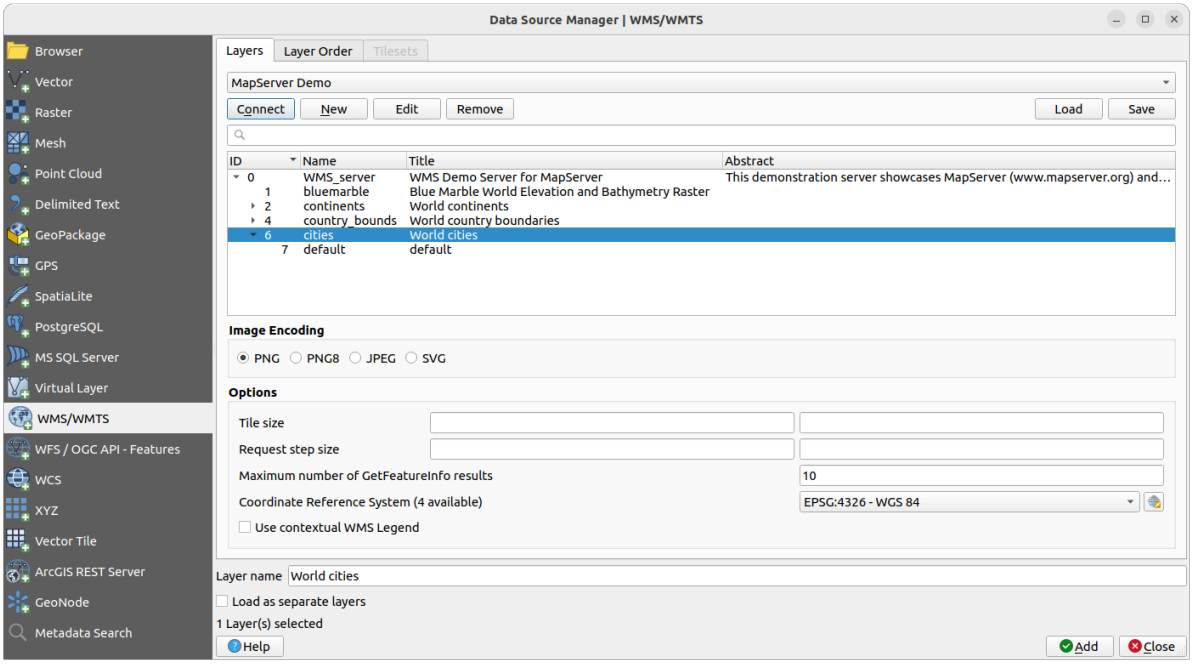


図 22.3: 利用可能なレイヤのフィルタを持った、WMS サーバーを追加するためのダイアログ

ダイアログの **レイヤ** タブの上部には、サーバーが提供する関連する画像スタイルでレイヤを埋め込むレイヤグループを含めることができるツリー構造が表示されます。各アイテムは、次の方法で識別できます：

- *ID*
- **名前**
- **タイトル**
- **要約**

リストは右上の角にある  ウィジェットを使ってフィルタできます。

イメージエンコーディング

イメージエンコーディング セクションは、クライアントとサーバーの両方でサポートされているフォーマットを示しています。画像の精度要件に応じていずれかを選択します。

Tip: イメージエンコーディング

典型的な WMS サーバーはイメージエンコーディングに JPEG か PNG を提案してくるでしょう。JPEG は損失のある圧縮形式ですが、PNG は生のラスタデータを忠実に再現します。

WMS データが自然の中での撮影であることを期待する場合、および/または画質の若干の損失を気にしない場合、JPEG を使用してください。このトレードオフは、一般的に PNG に比べてデータ転送要件を 5 倍減らします。

元のデータの正確な表現をしたい、そしてデータ転送の要件の増加を気にしない場合、PNG を使用してください。

オプション

ダイアログのオプション区域は WMS リクエストを設定する方法を提供します。次が定義できます:

- **タイルサイズ** WMS 要求を複数の要求に分割するためにタイルサイズを設定したい場合（例 256x256）
- **要求ステップサイズ**: タイルの境界でカットラベルの効果を減らしたい場合、ステップサイズを大きくすると、より大きなリクエスト、より少ないタイル、より少ないボーダーを作成します。デフォルト値は 2000 です。
- サーバーからの *GetFeatureInfo* **の最大値**
- 各 WMS レイヤは、WMS サーバーの機能に応じて、複数の CRS で表示できます。リストから WMS を選択すると、Web サーバーによって提供されるデフォルトの投影法のフィールドが表示されます。Select CRS を選択 ウィジェットを押して、WMS のデフォルトの投影法を WMS サーバーでサポートされている別の CRS に置き換えます。

  図 10.3 で示したものと同様のダイアログが表示されます。WMS 版のダイアログとの主な違いは、WMS サーバーでサポートされている CRS のみが表示されることです。
- 最後に、WMS サーバーがこの機能をサポートしている場合 **文脈** *WMS* **凡例を使用** を有効にできます。そのときは、現在の地図ビューの範囲に関連する凡例だけ表示されますので、現在の地図で見えないものの凡例項目は含まれません。

ダイアログの下部には、 **レイヤ名** というテキストフィールドがあり、選択した項目の **タイトル** を表示します。この名前は自由に変更することができます。この名前は、**追加** ボタンを押して QGIS にレイヤを読み込んだ後、**レイヤ** パネルに表示されます。

一度に複数のレイヤを選択することができますが、画像スタイルはレイヤ毎に 1 つのみです。複数のレイヤが選択されると、それらは WMS サーバーで結合され、1 つのレイヤとして QGIS に一括送信されます。デフォルトの名前は、スラッシュ (/) で区切られた元のタイトルのリストです。ただし、 **別のレイヤとして読み込む** を選択することができます。

レイヤ順序

レイヤ順序 タブは、現在接続している WMS サーバーから利用できる選択されたレイヤを一覧表示します。

サーバーによってレンダリングされた WMS レイヤは、 **レイヤ** タブにリストされている順序で、リストの上から下にオーバーレイされます。オーバーレイの順序を変更するのに、 **レイヤ順序** タブの **上** ボタンと **下** ボタンを使うことができます。

透過性

レイヤプロパティ の **グローバルな不透明度** 設定は、利用可能な場合は常にオンになるようにハードコードされています。

22.1.5 タイルセット

WMTS（キャッシュ WMS）サービスを使用する場合、サーバーが提供する **タイルセット** タブを閲覧することができます。タイルのサイズ、フォーマット、サポートされている CRS などの追加情報は、この表に記載されています。

```
# example of WMTS service

https://opencache.statkart.no/gatekeeper/gk/gk.open_wmts?service=WMTS&
↪request=GetCapabilities
```

読み込むレイヤーを選択すると、 *Interpretation method* を適用し、QGIS の通常の *raster renderers* を使ってスタイリングできる、シングルバンド float タイプのラスターレイヤーに変換することも可能です。

この機能と組み合わせて、タイルスケールスライダーを使用するには、 **ビュー　パネル**（または **設定　パネル**）を選択し、 **タイルスケールパネル** を選びます。これにより、タイルサーバーから利用可能なスケールが、素敵なスライダーとともにドッキングして表示されます。

22.1.6 地物特定ツールの利用

一度 WMS サーバーを追加してしまい、WMS サーバーからのすべてのレイヤが照会可能であるならば、識別 ツールを使用して地図キャンバス上でピクセルを選択できます。クエリが行われた各選択のための WMS サーバーに行われます。クエリの結果はテキスト形式で返されます。このテキストの書式は、使用される特定の WMS サーバーに依存しています。

フォーマット選択

複数の出力フォーマットがサーバーによってサポートされている場合、サポートされる形式とコンボボックスが自動的に識別結果ダイアログに追加され、選択されたフォーマットは、レイヤのためのプロジェクトに格納されてもよいです。

GML フォーマットサポート

:sup:`識別 `ツールは、GML 形式で WMS サーバーの応答（GetFeatureInfo）を（それはこの文脈では QGIS GUI での地物と呼ばれる）をサポートしています。「地物」フォーマットがサーバーによってサポートされており、選択された場合は、識別ツールの結果は、通常のベクタレイヤからのように、ベクタ地物です。単一の地物をツリーで選択された場合、それは地図で強調表示され、それがクリップボードにコピーされ、別のベクタレイヤに貼り付けできます。GML 形式で GetFeatureInfo をサポートするために、以下の UMN Mapserver の設定例を参照してください。

```
# in layer METADATA add which fields should be included and define geometry (example):

"gml_include_items"   "all"
"ows_geometries"      "mygeom"
"ows_mygeom_type"     "polygon"

# Then there are two possibilities/formats available, see a) and b):
```

(次のページに続く)

(前のページからの続き)

```
# a) basic (output is generated by Mapserver and does not contain XSD)
# in WEB METADATA define formats (example):
"wms_getfeatureinfo_formatlist" "application/vnd.ogc.gml,text/html"

# b) using OGR (output is generated by OGR, it is sent as multipart and contains XSD)
# in MAP define OUTPUTFORMAT (example):
OUTPUTFORMAT
    NAME "OGRGML"
    MIMETYPE "ogr/gml"
    DRIVER "OGR/GML"
    FORMATOPTION "FORM=multipart"
END

# in WEB METADATA define formats (example):
"wms_getfeatureinfo_formatlist" "OGRGML,text/html"
```

22.1.7 プロパティを表示する

WMS サーバーを追加したら、凡例でそのサーバーを右クリックして **プロパティ** を選択することで、そのプロパティを見ることができます。WMS/WMTS レイヤーのプロパティはラスターレイヤーのプロパティとよく似ているので、詳細な説明は **ラスタプロパティダイアログ** で見ることができます。しかし、いくつかの相違点があるので、以下に説明します。

情報プロパティ

メタデータタブ

タブ **メタデータ** は、一般的に、そのサーバーから返された機能の声明から収集され、WMS サーバーに関する豊富な情報を表示します。多くの定義が WMS の規格を読むことによって収集できますが（ **文献と** *Web* **参照** でオープン地理空間コンソーシアムを参照)、ここで便利ないくつかの定義は：

- **サーバープロパティ**
  - **WMS バージョン** --- サーバーによってサポートされた WMS のバージョン.
  - **画像フォーマット** --- サーバーが地図を描画するときに応答できる MIME-タイプのリスト。QGIS では Qt ライブラリが構築された基礎になるフォーマットは何でもサポートしています。通常は少なくとも 画像/png と 画像/jpeg があります。
  - **アイデンティティ形式** ---サーバーが識別ツールを使用する場合に対応できる MIME-タイプのリスト。現在、QGIS は``テキスト plain``タイプをサポートしています。
- **レイヤプロパティ**
  - **選択** ---そのサーバーは、このプロジェクトに追加したときに、このレイヤを選択したかどうか。

- **可視** ---このレイヤは（まだ QGIS のこのバージョンでは使用されない）、凡例に見えるように選択されているか否か。
- **特定可能** ---このレイヤは特定ツールが使用されたときに結果を返すかどうか。
- **透明可能** ---このレイヤが透明でレンダリングすることが可能かどうか。QGIS のこのバージョンでは、これが `Yes` であり画像符号化が透明度をサポートしている場合、常に透明度を使用します。
- **ズームイン可能** ---このレイヤは、サーバーによってズームインすることが可能かどうか。QGIS のこのバージョンでは、すべての WMS レイヤがこれを `Yes` に設定されることを前提としています。欠陥レイヤは奇妙にレンダリングされることがあります。
- **カスケードカウント** --- WMS サーバーは、レイヤのためのラスタデータを取得するために、他の WMS サーバーへのプロキシとして動作することができます。このエントリは、このレイヤのための要求は結果のために WMS サーバーをピアに転送された回数を示しています。
- **固定幅、固定高さ** ---このレイヤは、固定されたソースピクセル寸法を有しているか否か。QGIS のこのバージョンは、すべての WMS レイヤは、何もこのセットを前提としています。欠陥レイヤは奇妙にレンダリングされることがあります。
- **WGS 84 バウンディングボックス** --- WGS 84 座標での、レイヤのバウンディングボックス。一部の WMS サーバーはこれを正しく設定しません（例えば、代わりに UTM 座標が使われます）。この場合、このレイヤの初期ビューは、QGIS によって非常に「縮小」された外観でレンダリングされるかもしれません。WMS のウェブマスターに、WMS の XML 要素 `LatLonBoundingBox` 、 `EX_GeographicBoundingBox` または CRS:84 `BoundingBox` として知っている可能性がある、このエラーを知らされるべきです。
- **CRS で利用可能** ---このレイヤは、WMS サーバーによってレンダリングすることができることを予測。これらは、WMS-ネイティブフォーマットで記載されています。
- **スタイルで利用可能** ---このレイヤが WMS サーバーによって描画されることができる画像のスタイル。

時系列プロパティ

ラスタ **時系列プロパティ** （すなわち **動的時系列コントロール** ）は、WMS と WMTS レイヤに設定することができます。デフォルトでは、時間軸を有効にした WMS または WMTS レイヤがプロジェクトに追加されると、 **レイヤ** パネルに Temporal Layer というアイコンが横に表示されます。その *Temporal* プロパティのデフォルトは **自動** 時系列モードで、これはレイヤがデフォルトで 時系列コントローラー の現在の時間範囲に従うことを意味しています。

動的時系列コントロール のチェックを外し、**静的** *WMS-T* **時間範囲** のオプションを選択すると、レイヤの特定の静的時間値を表示することができます:

- *Server default*
- *Predefined date* を使用して連続しない時間範囲のデータを公開しているサーバを指定するか、*Predefined range* を使用して利用可能な日付範囲を公開しているサーバを指定します。後者の場合は *Start date* と *End date* が必須です。予想されるフォーマットは、プロバイダが連続した期間のデータを持っているかどうかに応じて、参照時間オプション（下記参照）から推測することができます。

- *Follow project's temporal range* as defined in the project's properties dialog

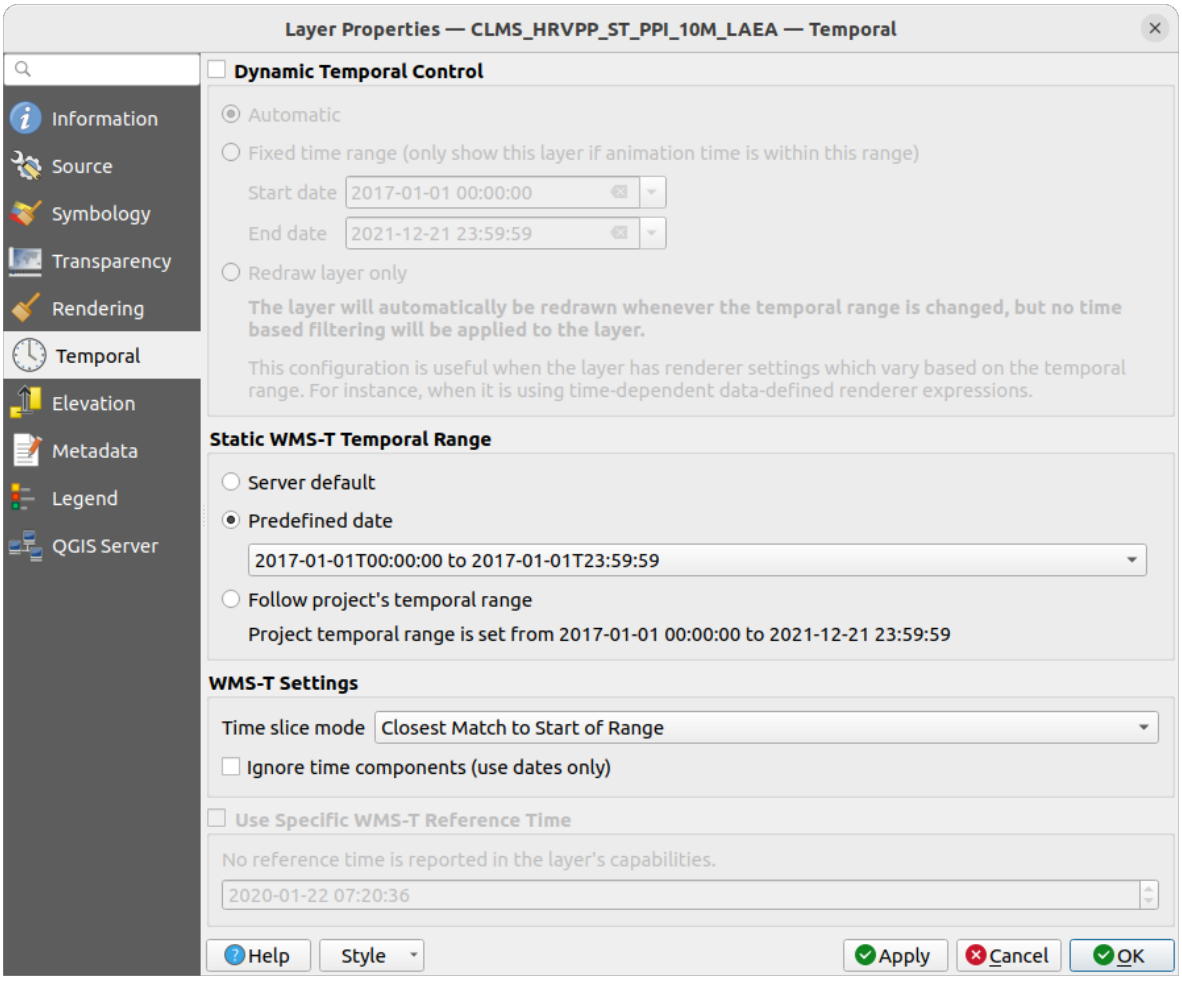


図 22.4: Temporal properties of a WMTS layer

Whatever temporal data control is in use, there are some *WMS-T Settings* to help display the correct temporal data:

- *Time slice mode* which can be:
  - *Use whole temporal range*
  - *Match to start of range*
  - *Match to end of range*
  - *Closest match to start of range*
  - *Closest match to end of range*
- *Ignore time components (use dates only)*: If checked, the time component of temporal queries will be discarded and only the date component will be used in server requests.

You can also *Use Specific WMS-T Reference Time* picked from times reported in the layer's capabilities. Convenient for servers which expose a non-contiguous set of date time instances (instead of a range of dates).

QGIS サーバープロパティ

ラスターレイヤー **プロパティ** に加えて、WMS/WMTS レイヤーを QGIS Server で公開すると、以下のオプションが表示されます:

- *WMS* **印刷レイヤ**: 印刷に使用する代替の WMS レイヤを設定することができます (GetProjectSettings 返信)。一般的に印刷に適さない WMTS レイヤに便利です。
- ☑ *WMS/SMTS* **データソース** *URI* **を公開する**: ウェブクライアントが WMS/WMTS データを直接取得できるようにします
- ☑ **背景レイヤ扱いする**

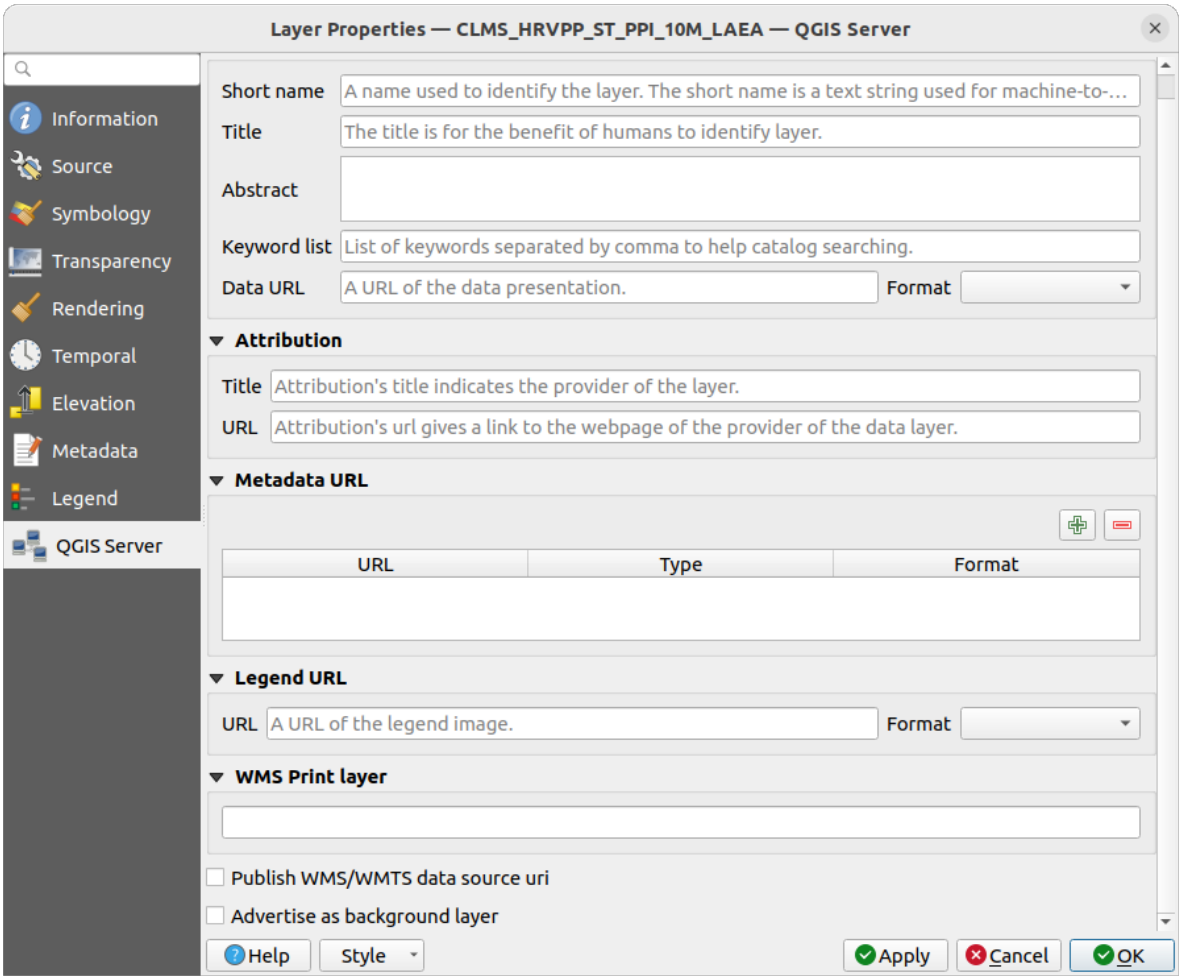


図 22.5: WMS/WMTS レイヤの QGIS サーバープロパティ

22.1.8 WMS 凡例のグラフィックをコンテンツのテーブルおよびレイアウトに表示する

QGIS WMS データプロバイダーでは、コンテンツレイヤリストのテーブルで、また印刷レイアウトで、凡例のグラフィックを表示できます。WMS の凡例は、WMS サーバーに GetLegendGraphic 機能があり、レイヤが指定された getCapability の URL を持っている場合にのみ表示されますので、さらにそのレイヤのためのスタイル設定を選択する必要があります。

legendGraphic が利用可能な場合、それはレイヤの下に表示されます。それは少しであり、(QgsLegendInterface アーキテクチャ上の制限のために）実際の次元でそれを開くために、それをクリックする必要があります。レイヤの凡例をクリックすると、フル解像度での凡例を持つフレームを開きます。

印刷レイアウトでは、凡例は元の（ダウンロードされた）寸法で統合されます。凡例グラフィックの解像度は、 **凡例**　 *WMS LegendGraphic* の下のアイテムプロパティで設定して、印刷要件に一致させることができます。

凡例は、現在の縮尺に基づいてコンテキスト情報を表示します。WMS の凡例は、WMS サーバーが GetLegendGraphic 機能とレイヤが指定 getCapability の URL を持っている場合にのみ表示されますので、スタイリングを選択する必要があります。

22.2 WCS クライアント

ウェブカバレッジサービス（WCS）は、科学的モデルへの入力として、および他のクライアントのために、クライアント側のレンダリングに役立つ形でラスタデータへのアクセスを提供します。WCS は WFS と WMS と比較できます。WMS と WFS サービスインスタンスとして、WCS は、クライアントが、空間的な制約や他のクエリー基準に基づいて、サーバーの情報保有の部分を選択できます。

QGIS ではネイティブ WCS プロバイダーを持っており、バージョン 1.0 および 1.1 の両方（大幅に異なっている）をサポートしていますが、1.1 には多くの問題がある（すなわち、各サーバーは、様々な特殊性と異なる方法でそれを実装する）ため、現在は 1.0 を好みます。

ネイティブ WCS プロバイダーは、すべてのネットワーク要求を処理し、すべての標準 QGIS のネットワーク設定（特にプロキシ）を使用しています。キャッシュ・モードを選択することも可能である（「常にキャッシュ」、「キャッシュを好む」、「ネットワークを好む」、「常にネットワーク」)、および時間的なドメインは、サーバーによって提供されている場合、プロバイダーはまた、時間位置の選択をサポートしています。

警告: **認証** タブで **ユーザー名** と **パスワード** を入力すると、保護されていない資格情報が接続構成に保持されます。たとえば、プロジェクトファイルを誰かと共有した場合、これらの **認証情報が見えてしまいます** 。したがって、代わりに **認証設定** （ **設定** タブ）に資格情報を保存することをお勧めします。詳細は **認証システム** 参照。

22.3 WFS および WFS-T クライアント

In QGIS, a WFS layer behaves pretty much like any other vector layer. You can identify and select features, and view the attribute table. QGIS supports WFS 1.0.0, 1.1.0, 2.0 and OGC API - Features (OAPIF), including editing (through WFS-T). QGIS also supports background download and progressive rendering, on-disk caching of downloaded features and version autodetection.

一般に、WFS レイヤの追加は、WMS で使用される手順と非常によく似ています。デフォルトのサーバーは定義されていないため、独自のサーバーを追加する必要があります。 WFS サーバーは、 *MetaSearch* **プラグイン** またはお気に入りの Web 検索エンジンを使用して見つけることができます。パブリック URL を含むリストは多数あり、維持されているものと維持されていないものがあります。

WFS レイヤを読み込む

例として、Gateway Geomatics WFS サーバーを使ってレイヤを表示します。

```
https://demo.gatewaygeomatics.com/cgi-bin/wfs_gateway?REQUEST=GetCapabilities&
↪VERSION=1.0.0&SERVICE=WFS
```

WFS レイヤを読み込めるように、まずその WFS サーバーへの接続を作ります。

1. 

を押して **データソースマネージャ** ダイアログを開きます
2.  *WFS/OGC API* - **地物** タブを有効にします
3. **新規**... をクリックして **新規** *WFS* **接続を作成** ダイアログを開きます
4. 名前に `Gateway Geomatics` を入力します
5. URL を入力します（上記参照）

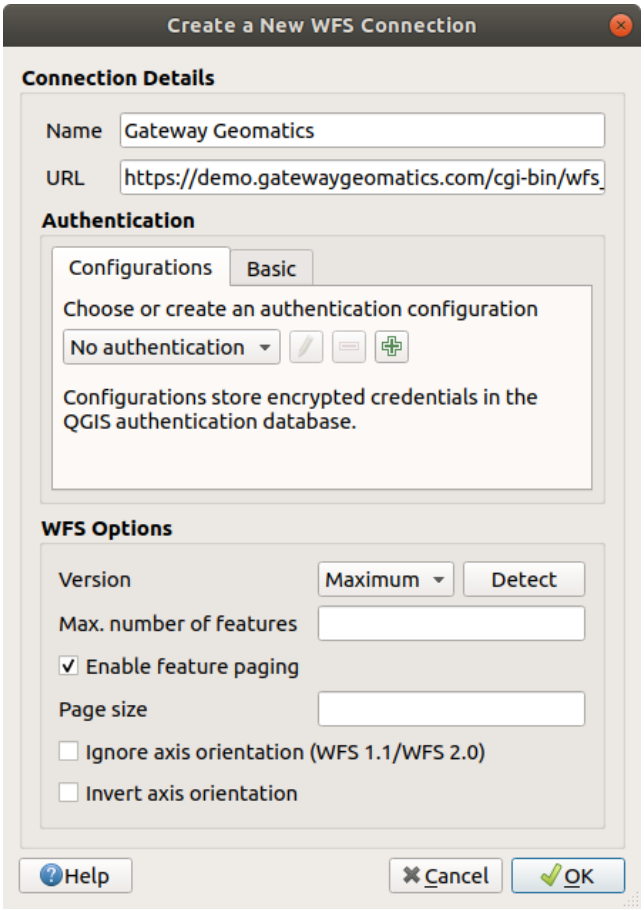


図 22.6: WMS サーバーへの接続を作る

注釈: OGC API - 地物 (OAPIF) の場合、与える URL は landing page 、つまり、そこからすべての利用可能なサービスのエンドポイントに移動可能なメインページでなければなりません。

---

6. WFS 設定ダイアログでは、以下の設定ができます：

   - サーバーの WFS バージョンを示します。不明な場合は **検出する** ボタンを押すと自動的に取得されます。
   - 1 回の GetFetFeature リクエストで取得する :guilabel:``地物数の最大値` を決めます。空の場合、制限は設定されません。
   - WFS バージョンに依って、次のどれかを指定します:
     - **地物ページングを有効にする**、そして、**ページサイズ** に取得する地物の最大数を指定します。限度を設定しない場合はサーバーのデフォルトが適用されます。
     - **軸方位を無視する** *(WFS 1.1/WFS 2.0)* を強制する
     - **軸方位を逆にする**。
     - **トランザクションに** *GML2* **を使う**。

> **警告:** **認証** タブで **ユーザー名** と **パスワード** を入力すると、保護されていない資格情報が接続構成に保持されます。たとえば、プロジェクトファイルを誰かと共有した場合、これらの **認証情報が見えてしまいます** 。したがって、代わりに **認証設定** （ **設定** タブ）に資格情報を保存することをお勧めします。詳細は **認証システム** 参照。

7. *OK* を押して接続を作成します。

好みで設定した可能性のあるすべてのプロキシ設定も認識されていることに注意してください。

これで、上記の接続から WFS レイヤを読み込む準備ができました。

1. **サーバ接続** ドロップダウンリストから 'Gateway Geomatics' を選択します。
2. **接続** をクリックします
3. リストの:guilabel:*Parks* レイヤを選択します
4. また次を選択することもできます:
   - **タイトルをレイヤ名にする** は、サーバー上で定義されたレイヤの **名前** の代わりにタイトルを **レイヤ** パネルに表示します。
   - **ビュー領域に重なる地物のみをリクエストする**
   - レイヤの CRS をサーバーがサポートしている他のものに **変更**... します
   - サービスから取得する特定の地物を指定するためのクエリを作ります: レイヤの行をダブルクリックすると *SQL* **クエリコンポーザ** ダイアログが開きます。このダイアログは、ソートやフィルタリング、SQL 関数、空間述語、演算子の束を使い、サービスの利用可能なテーブルとカラムに依存した、高度な SQL クエリを記述するためのウィジェットを提供します。

     作成したクエリは、検証後に *WFS / OGC API* - **地物** テーブルの *SQL* カラムに表示され、フィルタリングされたレイヤは *Layers* パネルでその横に アイコンが表示されます。このように、いつでもクエリを調整することができます。
5. **追加** をクリックしてマップにレイヤを追加します。

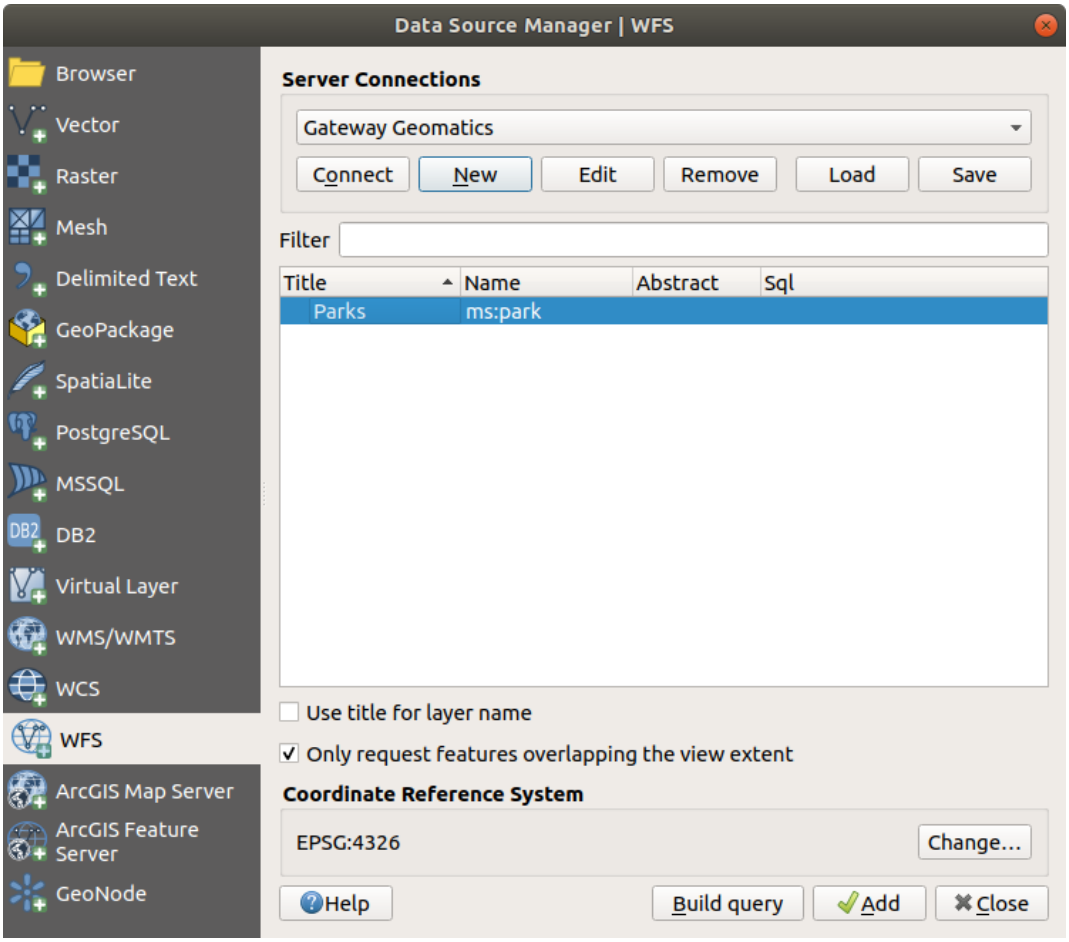


図 22.7: WFS レイヤを追加

ダウンロードの進行状況が QGIS メインウィンドウの左下に視覚化されていることに気付くでしょう。そのレイヤが読み込まれると、いくつかの地物を識別して選択し、属性テーブルを表示できます。

第23章　GPSデータの操作

23.1 GNSS/GPSデータの導入

23.1.1 GPSとは？

GPS、**全地球測位システム**、は、衛星型のシステムであり、GPS受信機さえあれば、世界中どこでも自分の正確な位置を知ることができます。GPSは航法の補助として、例えば飛行機やボート、ハイカーなどに利用されています。GPS受信機は、衛星からの信号を使って緯度、経度、（場合によっては）標高を算出します。ほとんどの受信機は次を保存する機能も持っています：

- 場所（**ウェイポイント**として知られています）
- 計画した**経路**を構成する場所の順序
- そして受信機の異動の経時的な**トラック**ログ。

ウェイポイント、経路及びトラックは、GPSデータにおける三つの基本的な地物型です。QGISはウェイポイントをポイントレイヤに表示し、経路とトラックはラインストリングレイヤに表示されます。

注釈: QGISはGNSS受信機もサポートしています。しかし、この文書の作成ではGPSという用語を使っています。

23.1.2 GPSのデバイス型を決める

GPSデバイスには様々な種類があります。QGISでは**設定**　**オプション**　*GPS*　*GPSBabel*タブで独自のデバイス型を定義し、使用するパラメータを設定することができます。詳しくは*GPSBabel*を参照してください。

いったん新しいデバイス型を作成すると, ダウンロードとアップロードツールのデバイスリストに表示されます.

23.1.3 GPS データを転送又は読み込む

GPS データを格納するための異なるファイル形式はいくつもあります。同じファイル内のウェイポイント、ルートとトラックの任意の数を含むことができ、標準の交換フォーマットである QGIS が使用するフォーマット GPX（GPS 交換フォーマット）と呼ばれます。

GPX ファイルを読み込むには：

1. **データソースマネージャ** ダイアログの *GPS* タブを開きます、つまり：
   - ツールバーの  データソースマネージャを開く ボタンをクリックし（または Ctrl+L を押し）、目標のタブを有効にします
   - 又は、**レイヤ** **レイヤを追加**  *GPX* **レイヤを追加**... を選択します
2. *GPX* **データセット** オプションの隣にある ... ブラウズ ボタンを使って GPX ファイルを選びます
3. ファイルから読み込む **地物の型** を選ぶにはチェックボックスを使います。それぞれの地物の型（**ウェイポイント**、**トラック** 又は **ルート**）は別々のレイヤに読み込まれます。

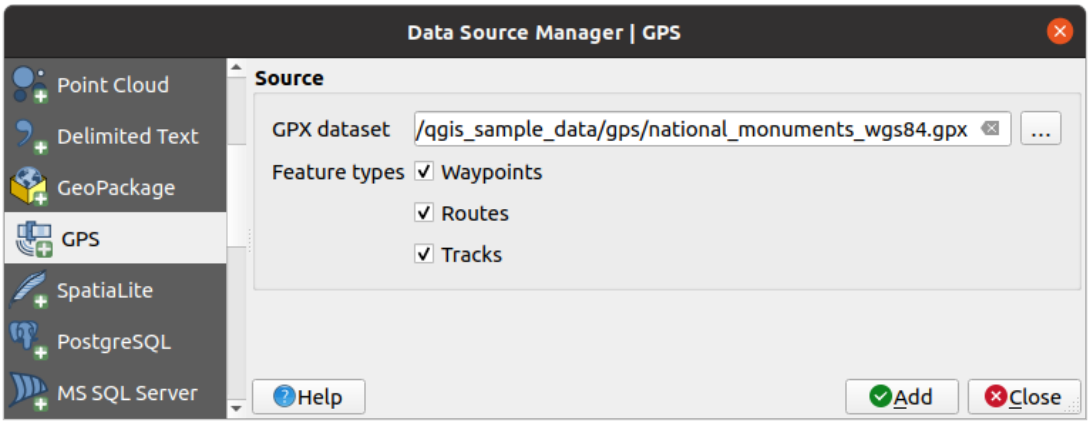


図 23.1: GPS データの読み込みダイアログ

QGIS は GPX ファイルを使うので、他の GPS ファイル形式を GPX に変換する方法が必要です。多くの形式はフリーのプログラム `GPSBabel <https://www.gpsbabel.org>`_ を使ってこれを行うことができます。このプログラムはコンピュータと GPS デバイス間で GPS データを転送することもできます。QGIS はこれらの処理を GPSBabel に依存し、*GPS* **グループ** から利用できる便利なプロセシングアルゴリズムを提供しています。

注釈: GPS ユニットでは、異なる座標系でデータを保存できます。GPX ファイルを（GPS ユニットまたはウェブサイトから）ダウンロードし、QGIS で読み込む場合は、GPX ファイルに保存されているデータが WGS 84（緯度経度）を使用していることを確認してください。QGIS はこれを期待しており、公式な GPX 仕様です。GPX 1.1 Schema Documentation <https://www.topografix.com/GPX/1/1/>`_ を参照してください。

23.2 ライブ GPS 追跡

QGIS でライブ GPS 追跡を有効にするには、 **ビュー** **パネル** *GPS* **情報** を選択するか、 `Ctrl+0` を押す必要があります。すると、キャンバスの左側に新しいドッキングウィンドウが表示されます。

GPS 追跡ウィンドウには 3 つのスクリーンがあります:

- 位置 ： GPS による位置座標と、頂点や地物を手動で入力するためのインターフェース
- 信号: 衛星接続の信号強度
- オプション: GPS オプション画面（図 23.4 参照）

接続された GPS 受信機（オペレーティングシステムでサポートされている必要があります）で、**接続** をクリックするだけで、GPS を QGIS に接続します。2 回目のクリック（今度は **切断**）で、GPS 受信機はコンピュータから切断されます。GNU/Linux の場合、ほとんどの GPS 受信機との接続をサポートするために gpsd サポートが統合されています。したがって、QGIS を接続するためには、まず gpsd を適切に設定する必要があります。

:guilabel:`再センタリング`ボタンを押すと、地図は現在の GPS 位置にジャンプします。

> **警告:** キャンバスに自分の位置を記録したい場合は、最初に新しいベクタレイヤを作成し、トラックを記録できるようにステータスを編集可能に切り替えなければいけません。

GPS デバイスが接続され、ユーザーがマップキャンバス上でカーソルを動かすと、ライブステータスバーメッセージに、カーソルから GPS 位置までの距離と方位が表示されます。この表示では、プロジェクトの距離と方位の設定が尊重されます。

Tip: タッチスクリーンデバイス

タッチスクリーンデバイスでは、タップアンドホールドイベントを使用してライブステータスバーメッセージをトリガーします。

23.2.1 位置と追加属性

GPS が衛星からの信号を受信している場合、自分の位置が緯度、経度、高度に加え、追加属性で表示されます。

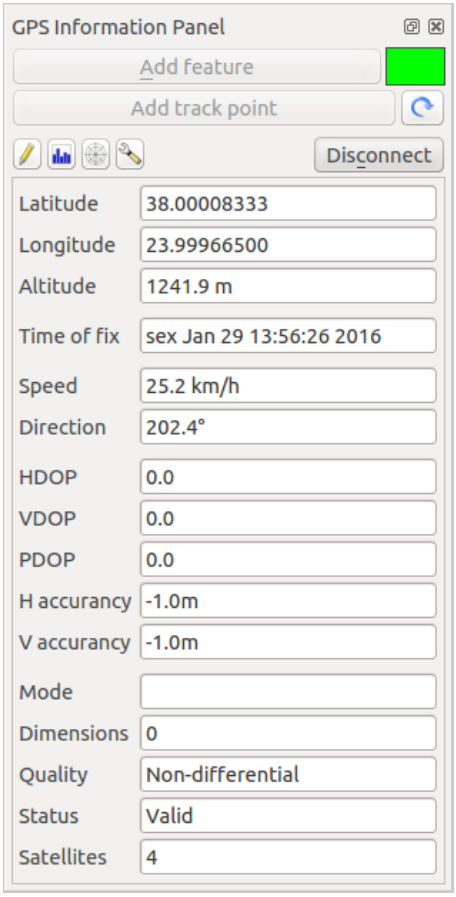


図 23.2: GPS 追跡位置と追加属性

23.2.2 GPS 信号強度

 ここでは、信号を受信して　　いる衛星の信号強度を確認できます。

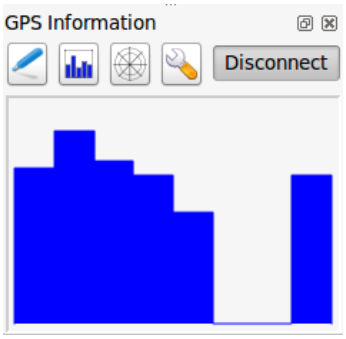


図 23.3: GPS 追跡信号強度

23.2.3 GPS オプション

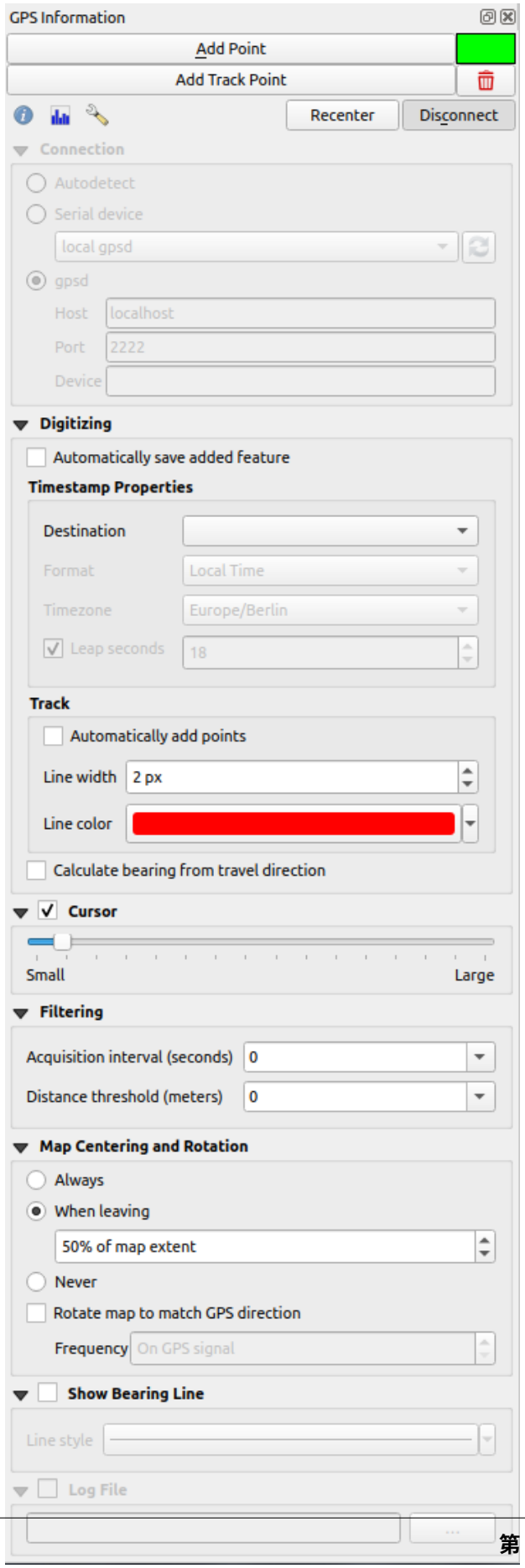


図 23.4: GPS 追跡オプションウィンドウ

ここでは次を指定することができます:

- **接続**
  - 接続に問題が生じた場合、切り替えることができます:
    - **自動検出**
    - **シリアルデバイス** （新しい GPS デバイスが接続された場合、再読み込みが必要）
    - *gpsd* （GPS が接続されるホスト、ポートおよびデバイスを選択）
  - 再び **接続** をクリックすると、GPS 受信機への接続が開始されます。
- **デジタイジング**
  - 編集モードになっている場合 **追加された地物を自動的に保存する** オプションを有効にすることができます. また **自動的に点を追加する** オプションを有効にして地図キャンパスに指定した幅と色で点を追加できます.
  - **移動方向の方位を計算する** は、デバイスが誤った方位測定を報告した場合に使用でき、記録された前の 2 つの位置に基づいて GPS 方位を計算します。
- **カーソル**: スライダー を使って、キャンバス上の位置カーソルを縮めたり大きくしたりすることができます。
- **フィルタ適用**: また、**取得間隔（秒）** と **距離閾値（メートル）** パラメータを設定して、受信機が静止している状態でもカーソルをアクティブにしておくことができます。
- **地図のセンタリングと回転**
  - **地図** *centering* を有効にすると、キャンバスが更新される方法を決定できます。これには「つねに」、「去り際」記録座標がキャンバスの外に移動を開始した場合、または「しない」地図範囲を維持する、があります。
  - *GPS* **の方角に合わせて地図を回転** を有効にすると、マップキャンバスが GPS の方位と同じ方向になるように自動的に回転されます。
- **方位線を表示** を有効にすると、GPS の位置から現在のパス方向を指す線が表示されます。
- 最後に **ログファイル** を有効にして、GPS 追跡で記録されたログメッセージを書き込むファイルのパスを指定できます。

手動で機能を設定したい場合は、位置 に戻り、 **点を追加** または **トラック点を追加** をクリックする必要があります。

23.2.4 ライブトラッキングの Bluetooth GPS への接続

QGIS を使用すると、フィールドデータ収集用の Bluetooth GPS を接続できます。このタスクを実行するには、GPS、Bluetooth デバイスとコンピュータの Bluetooth 受信機が必要です。

最初にあなたの GPS デバイスが認識され、コンピュータにペアリングさせる必要があります。 GPS をオンにし、あなたの通知領域に Bluetooth アイコンに移動し、新しいデバイスを検索します。

デバイス選択マスクの右側で、すべてのデバイスが選択されていることを確認し、GPS ユニットが利用可能なものの中に表示されるようにします。次のステップでは、シリアル接続サービスが利用できるはずなので、それを選択して *Configure* ボタンをクリックします。

Bluetooth の特性によって生じた、GPS 接続に割り当てられた COM ポートの番号を覚えておいてください。

GPS が認識された後、接続のためのペアリングを行います。通常、認証コードは `0000` です。

次に *GPS* **情報** パネルを開き、🔧GPS オプション画面に切り替えます。GPS 接続に割り当てられた COM ポートを選択し、**接続** をクリックします。しばらくすると、自分の位置を示すカーソルが表示されるはずです。

QGIS が GPS データを受信できない場合は、GPS デバイスを再起動して 5〜10 秒ほど待ってから、再度接続を試みてください。通常、このソリューションで対応できます。再び接続エラーを受信した場合、同じ GPS ユニットと対になった別の Bluetooth 受信機が近くにないことを確認してください。

23.2.5 GPSMAP 60cs を使用する

MS Windows

最も簡単に動作させる方法は、GPSGate というミドルウェア（フリーウェア、オープンではない）を使うことです。

プログラムを起動し、GPS デバイスをスキャンさせ（USB と BT の両方で動作します）、QGIS で、**自動検出** `モードを使って、ライブトラッキングパネルの *:guilabel:* **接続** をクリックするだけです。

Ubuntu/Mint GNU/Linux

Windows にとって簡単な手段は中間にサーバーを使うことであり、この場合は GPSD で、このため、

```
sudo apt install gpsd
```

それから `garmin_gps` カーネルモジュールを読み込みます

```
sudo modprobe garmin_gps
```

そして、GPS ユニットを接続します。接続したら、 `dmesg` を使って、GPS ユニットが実際に使っているデバイス (たとえば `/dev/ttyUSB0`) を確認します。これで gpsd を起動することができるようになりました。

```
gpsd /dev/ttyUSB0
```

最終的に QGIS ライブ追跡ツールで接続します。

23.2.6 BTGP-38KM データロガー (Bluetooth のみ) を使用する

GPSD (Linux) または GPSGate (Windows) を使うと手間が省略できます。

23.2.7 BlueMax GPS-4044 データロガー (BT と USB 両方) を使用する

MS Windows

ライブ追跡は USB と BT モードのどちらでも動作します、単に GPSGate を使用、あるいはそれなしでも、 **自動検出** モードを使用するか、またはツールに右ポートを指してください。

Ubuntu/Mint GNU/Linux

USB

ライブ追跡は GPSD ありでも動作しますし

```
gpsd /dev/ttyACM3
```

またはそれなしでも、直接デバイス（例えば `/dev/ttyACM3` ）に QGIS ライブ追跡ツールを接続することによって動作します。

Bluetooth

ライブ追跡は GPSD ありでも動作しますし

```
gpsd /dev/rfcomm0
```

またはそれなしでも、直接デバイス（例えば `/dev/rfcomm0` ）に QGIS ライブ追跡ツールを接続することによって動作します。

第24章　認証システム

24.1 認証システムの概要

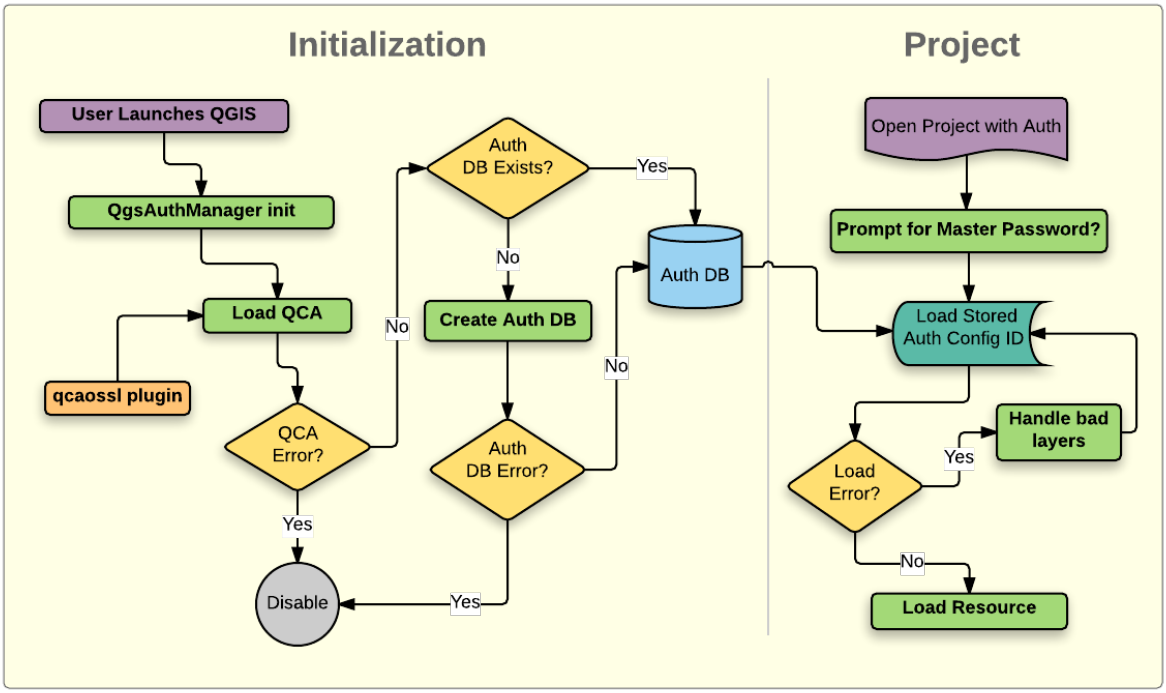


図 24.1: 認証システムの構造

24.1.1 認証データベース

新しい認証システムは、デフォルトでは <profile directory>/qgis-auth.db にある SQLite データベースファイルに認証設定を保存します。

この認証データベースは、それは通常の QGIS 設定から完全に分離されているので、他の現在の QGIS ユーザーの好みに影響を与えることなく、QGIS のインストール間で移動できます。最初にデータベースに設定を格納する際に、構成 ID（ランダム 7 文字の英数字文字列）が生成されます。これは設定を表し、これにより ID がその関連する資格情報の開示することなく、プレーンテキストアプリケーションコンポーネント（例えば、プロジェクト、プラグイン、または設定ファイルなど）に格納することを可能にします。

注釈: *qgis-auth.db* の親ディレクトリは、以下の環境変数、 QGIS_AUTH_DB_DIR_PATH を使用して設定、または --authdbdirectory オプションでの起動時にコマンドラインで設定できます。

24.1.2 マスターパスワード

データベース内に機密情報を保存したりアクセスしたりするには、ユーザは **マスターパスワード** を定義しなければなりません。暗号化されたデータをデータベースに初めて保存する際には、新しいマスターパスワードが要求され、確認されます。機密情報がアクセスされるとき、ユーザーはマスターパスワードの入力を求められます。その後、ユーザがキャッシュされた値をクリアするアクションを手動で選択しない限り、パスワードはセッションの終わりまで（アプリケーションが終了するまで）キャッシュされます。既存の認証設定を選択するときや、サーバー構成に設定を適用するとき（WMS レイヤを追加するときなど）など、認証システムを使用するインスタンスによっては、マスターパスワードの入力を必要としません。

パスワードをあなたのコンピュータの `Wallet/Keyring` に保存することもできます。

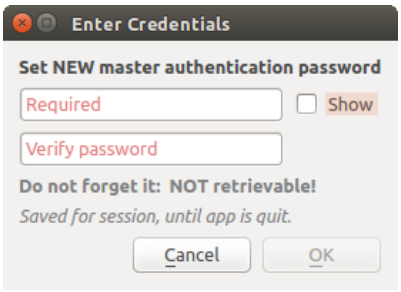


図 24.2: 新しいマスターパスワードの入力

注釈: マスターパスワードを含むファイルへのパスは、以下の環境変数、 `QGIS_AUTH_PASSWORD_FILE` を使用して設定できます。

マスターパスワードの管理

一度設定すると、マスターパスワードはリセットできます。現在のマスターパスワードはリセットする前に必要になるでしょう。このプロセスの間には、現在のデータベースの完全なバックアップを作成するオプションがあります。

図 24.3: マスターパスワードのリセット

ユーザーがマスターパスワードを忘れた場合は、それを取得したり、上書きする方法はありません。マスターパスワードを知らずに暗号化された情報を検索する手立てもありません。

ユーザーが既存のパスワードを誤って 3 回入力すると、ダイアログがデータベースを消去しようとします。

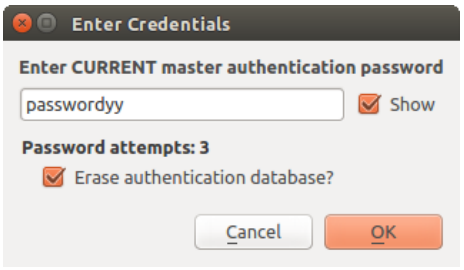


図 24.4: 3 つの無効な試みの後にパスワードのプロンプト

24.1.3 認証設定

認証設定は QGIS オプションダイアログ（**設定** **オプション**）の **認証** タブ中の **設定** から管理できます。

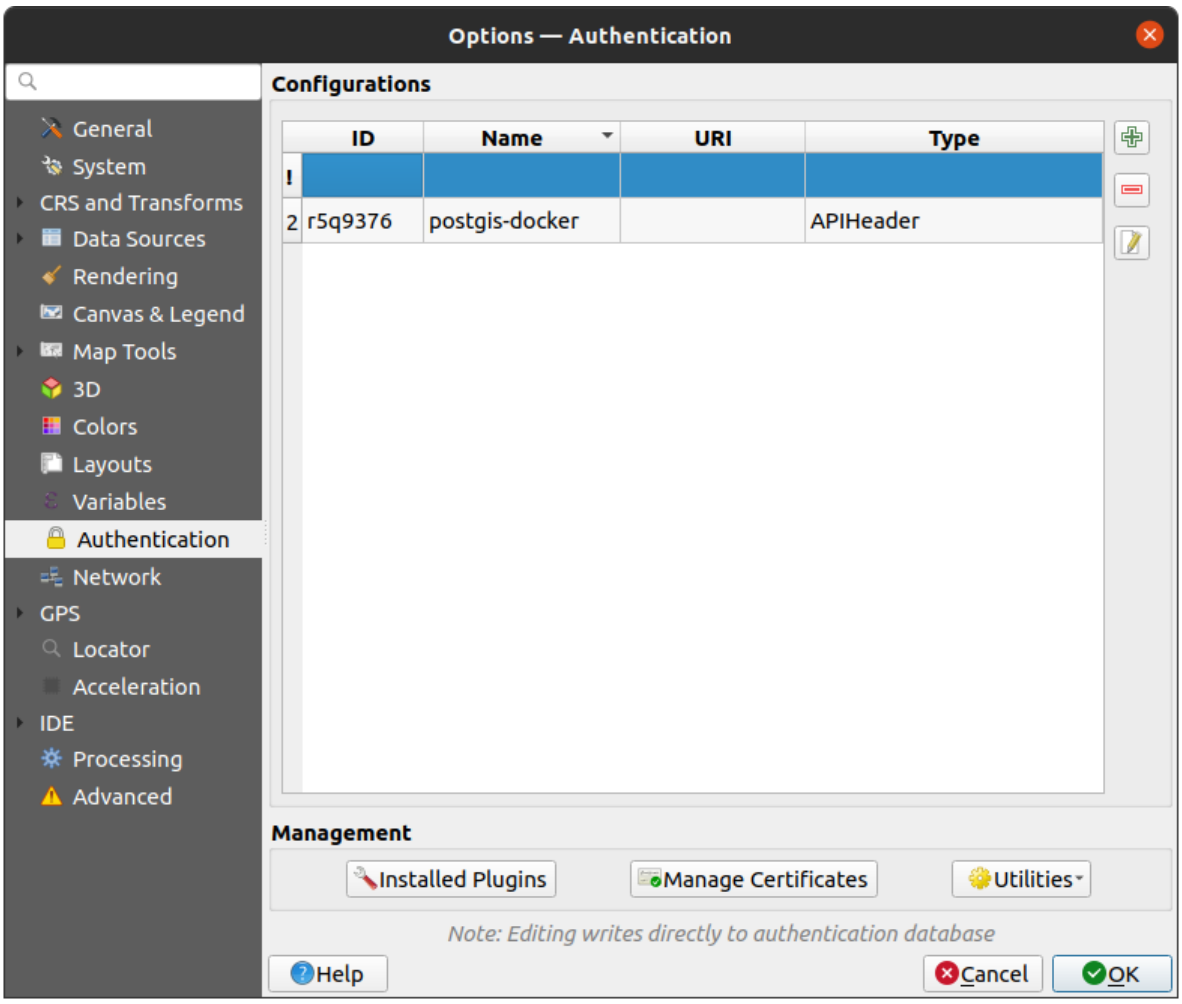


図 24.5: 設定エディタ

新しい設定を追加する ボタン、設定を削除する ボタン、および既存のものを変更する ボタンを使用できます。

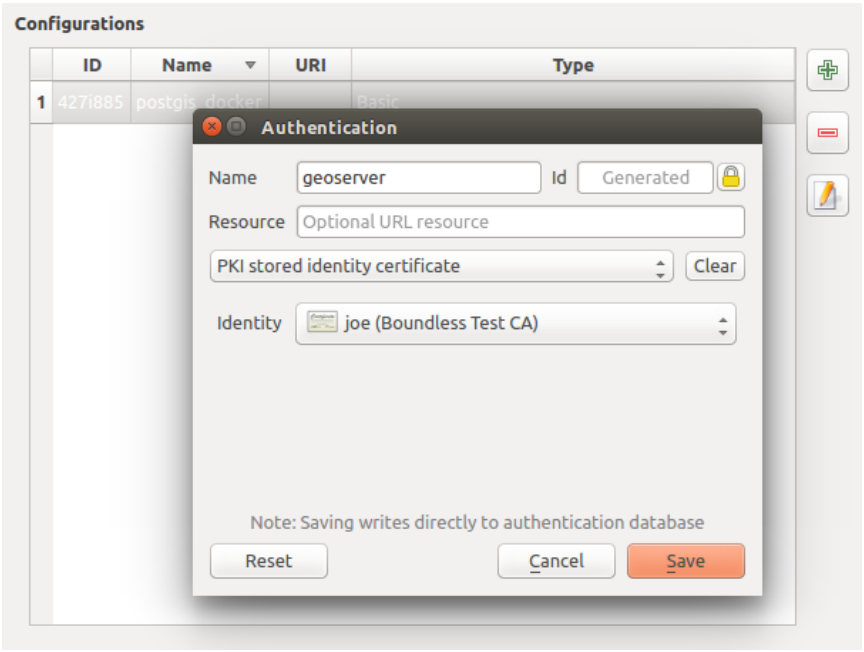


図 24.6: 設定エディタから設定を追加

認証設定管理のための同じタイプの操作（追加、編集、削除）は、OWS サービス接続を設定するように、所与のサービス接続を設定するときに行うことができます。そのため、認証データベース内で見つかった設定を完全に管理するためのアクションボタンが構成セレクタ内にあります。この場合、より包括的な構成管理を行う必要がない限り、QGIS オプションの **認証** タブ中の **設定** に行く必要はありません。

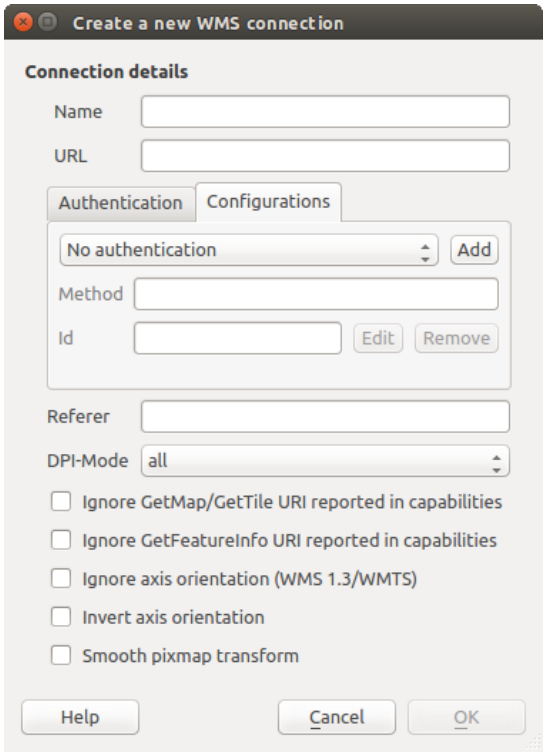


図 24.7: **新規追加**、**編集**、**削除** 認証設定ボタンを示す WMS 接続ダイアログ

認証設定を作成または編集するときに、必要な情報には、名前、認証方法および認証方法が必要であるこ

と、他の情報（利用可能な認証の種類についての詳細は **認証方式** を参照）。

24.1.4 認証方式

利用可能な認証は、データプロバイダプラグインが QGIS でサポートされているのと同じように、C++ プラグインによって提供されます。選択できる認証方法は、リソース/プロバイダに必要なアクセス（HTTP(S) やデータベースなど）、および QGIS コードとプラグインの両方にサポートがあるかどうかに関係します。そのため、認証方法プラグインによっては、認証設定セレクタが表示されていても適用できない場合があります。利用可能な認証方法プラグインとその対応リソース/プロバイダの一覧は **設定** **オプション** にアクセスし、 **認証** タブの **インストール済みプラグイン** ボタンをクリックすると表示されます。

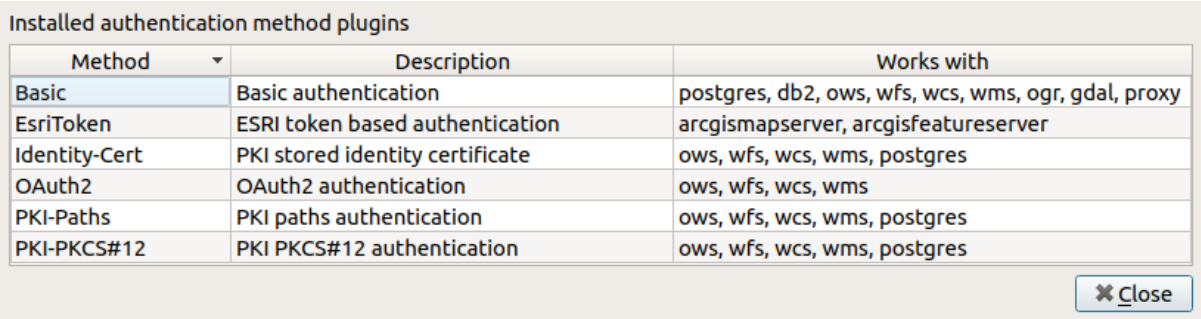


図 24.8: 利用可能なメソッドプラグイン一覧

プラグインは、QGIS の再コンパイルを必要とせずに新しい認証方法用に作成できます。現在、プラグインのサポートは C++のみであるため、ドロップインした新しいプラグインをユーザーが使用できるようにするには、QGIS を再起動する必要があります。プラグインを既存のターゲットインストールに追加する場合は、プラグインが QGIS の同じターゲットバージョンに対してコンパイルされていることを確認してください。

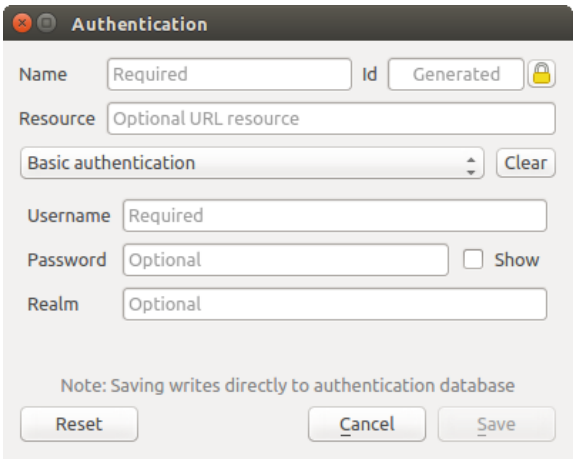


図 24.9: 基本 HTTP 認証構成

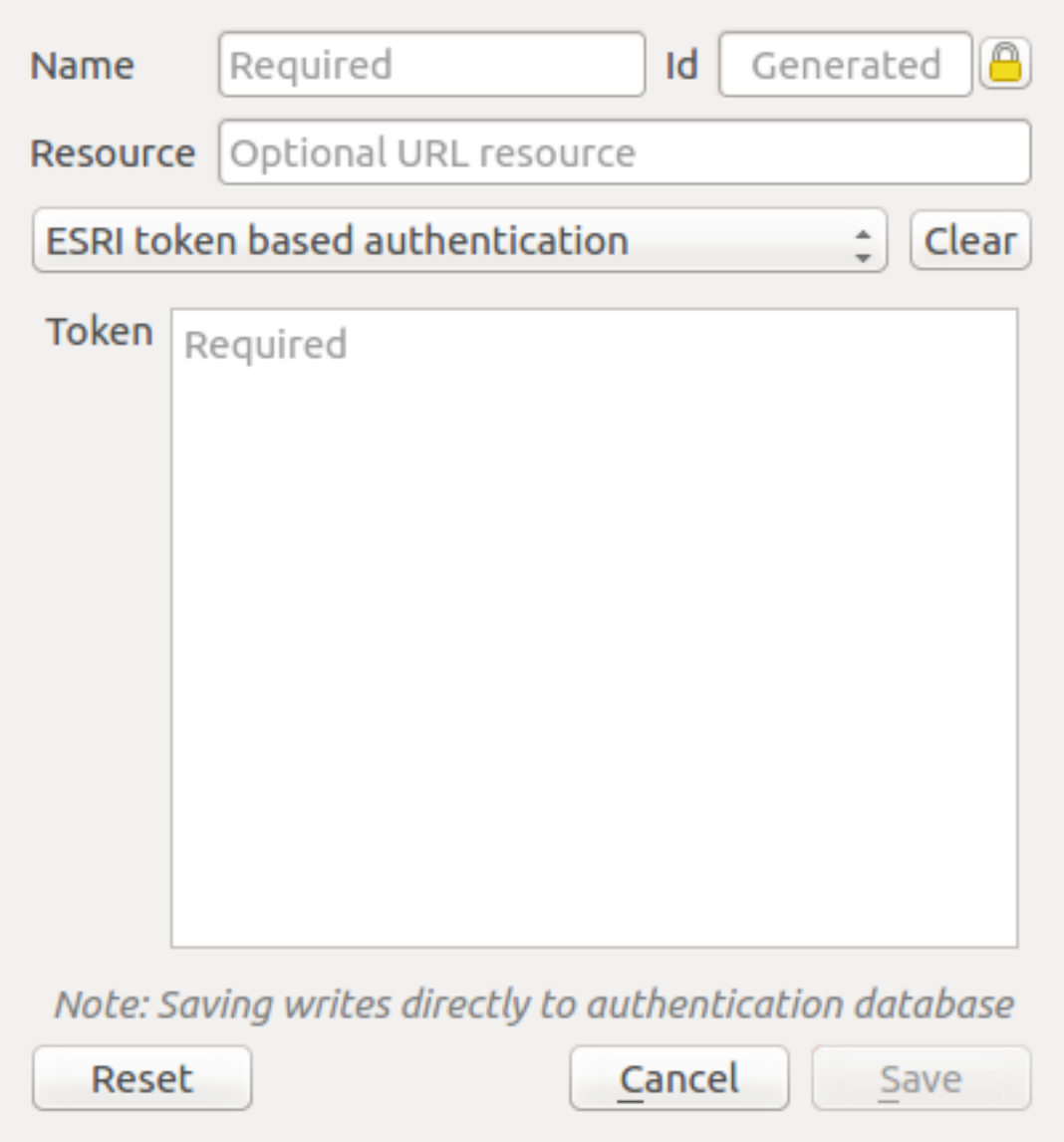


図 24.10: ESRI トークン認証構成

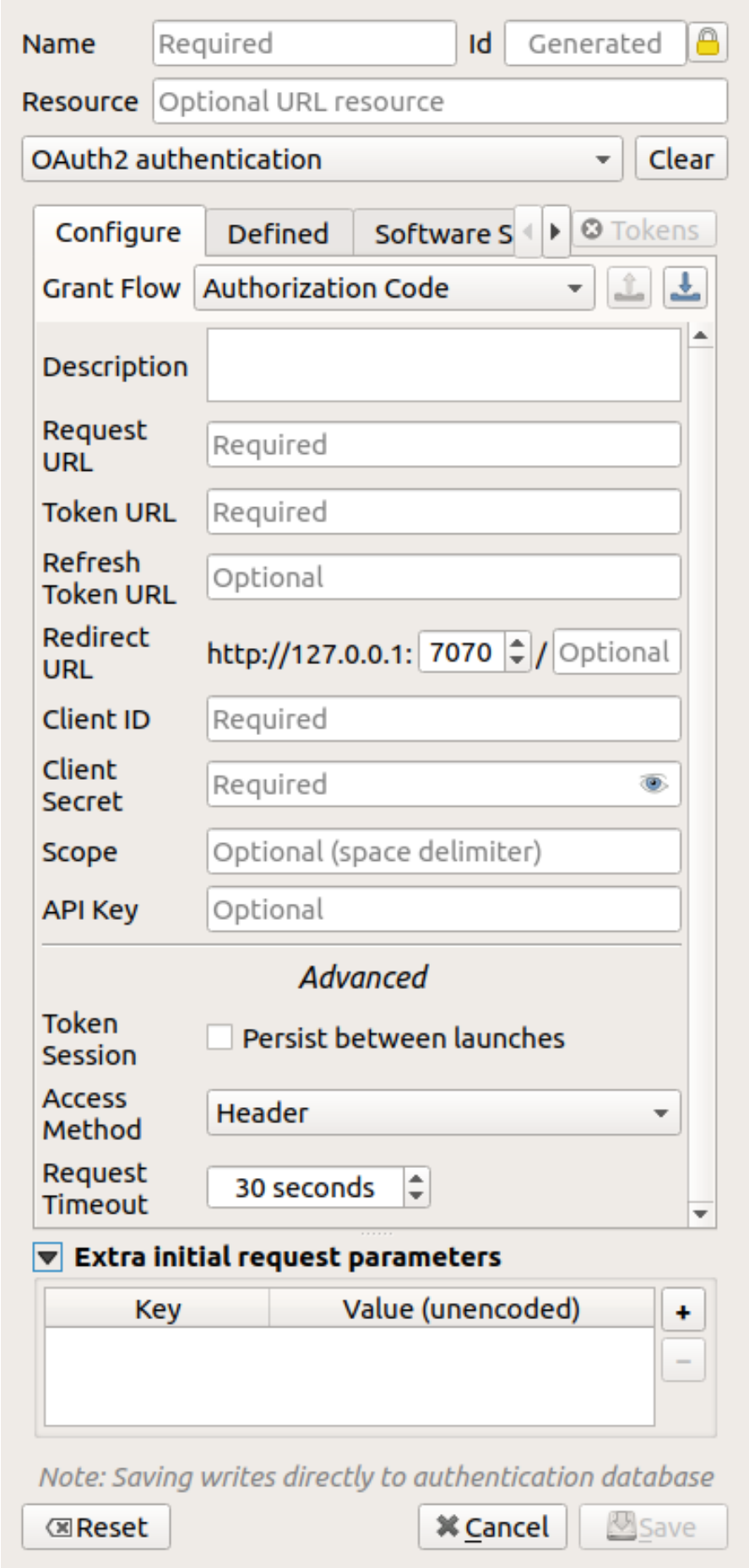


図 24.11: OAuth2 認証構成

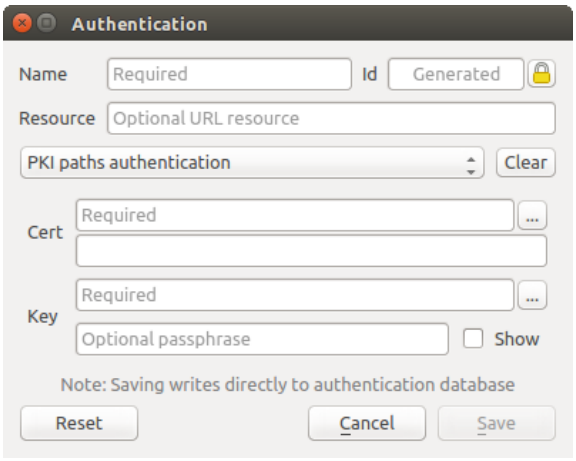


図 24.12: PKI パス認証構成

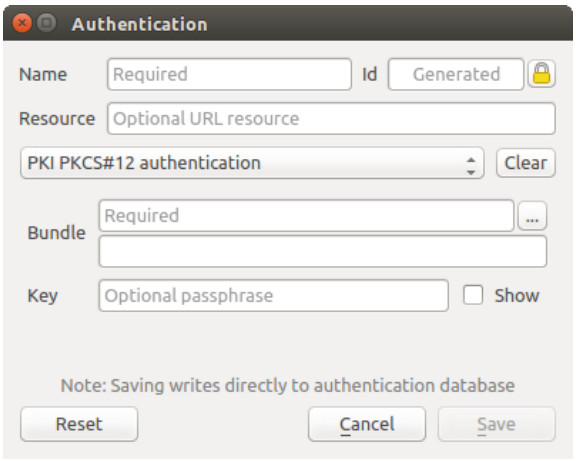


図 24.13: PKI PKCS ＃ 12 ファイルのパス認証構成

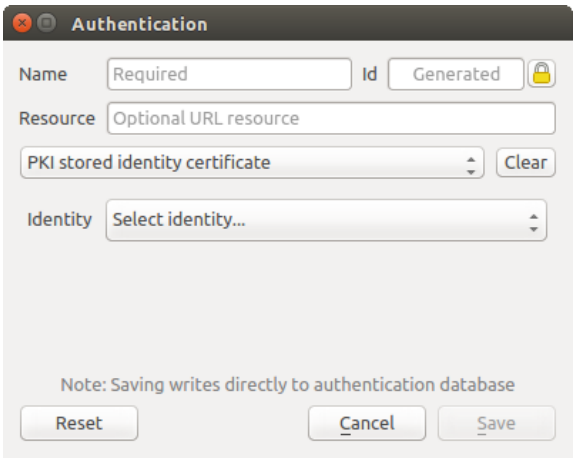


図 24.14: 保存されたアイデンティティ認証構成

注釈: リソースの URL は現在 **未実装** の機能で、最終的には指定された URL にあるリソースへの接続時

に特定の構成が自動選択できるようになります。

24.1.5 マスターパスワードと認証構成ユーティリティ

（**設定**　**オプション**）オプションメニューの下にある **認証** タブ、認証データベースと構成を管理するには、いくつかのユーティリティアクションがあります。

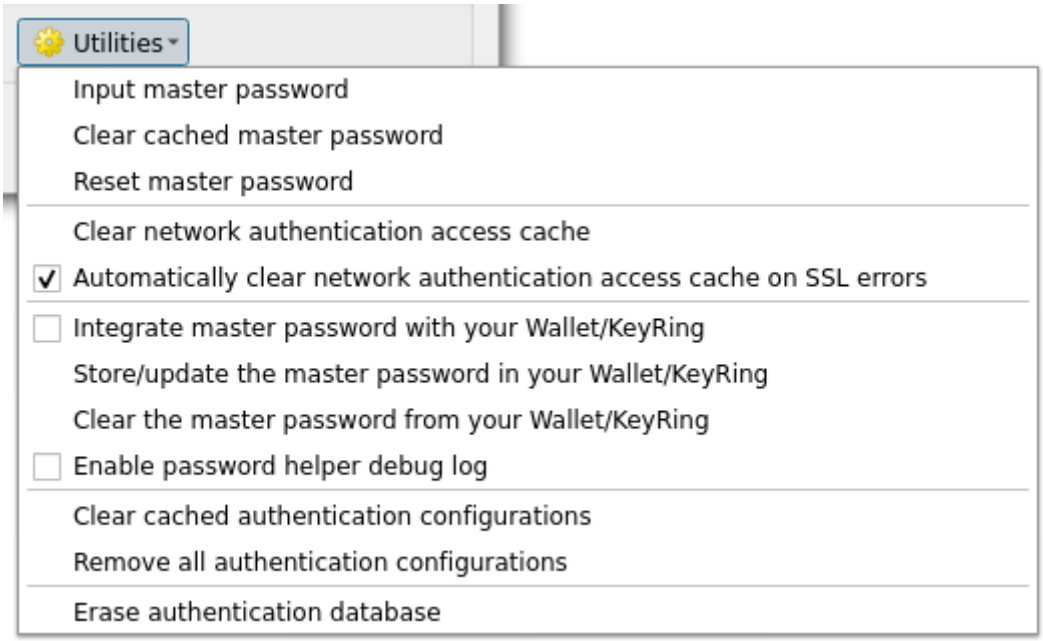


図 24.15: ユーティリティメニュー

- **マスターパスワードを入力する**: 認証データベースコマンドの実行とは無関係に、マスターパスワードの入力ダイアログを開きます
- **キャッシュされたマスターパスワードを消去する**: マスターパスワードがセットされているときにそれを外します
- **マスターパスワードをリセットする**: マスターパスワードを変更するダイアログを開き（現在のパスワードが判っている必要があります）、オプションで現在のデータベースをバックアップします
- **ネットワーク認証アクセスキャッシュをクリアする**: すべての接続の認証キャッシュをクリアします
- **SSL エラーでネットワーク認証アクセスキャッシュを自動的にクリアする**：接続キャッシュは、接続に失敗した場合にも、接続のすべての認証データを保存します。認証設定や認証局を変更した場合は、認証キャッシュをクリアするか、QGIS を再起動する必要があります。このオプションをオンにすると、SSL エラーが発生し、接続を中止することを選択するたびに、認証キャッシュが自動的にクリアされます。
- **あなたの Wallet/Keyring とマスターパスワードを統合する**: マスターパスワードを個人の Wallet/Keyring に追加します
- **Wallet/Keyring のマスターパスワードを保存更新**: 変更したマスターパスワードを Wallet/Keyring に保存します

- **あなたの Wallet/Keyring からマスターパスワードをクリアする**：Wallet/Keyring からマスターパスワードを削除します
- **パスワードヘルパーデバッグログを有効にする**: 認証方法のすべてのログ情報を格納するデバッグツールを有効にします
- **キャッシュされた認証構成をクリアする**: ネットワーク接続を高速化するために使用される設定の内部キャッシュをクリアします。これは QGIS のコアネットワークアクセスマネージャのキャッシュをクリアしません。それには QGIS の再起動が必要です。
- **すべての認証設定を削除する**: すべての設定レコードのデータベースをクリアします。その他のレコードは削除されません。
- **認証データベースを消去する**: 現在のデータベースのバックアップと、データベーステーブル構造の完全な再構築をスケジュールします。このアクションは、プロジェクトの読み込みのような、他の操作が一時的にデータベースが見つからないために中断やエラーが起きないよう、後の時間にスケジュールされます。

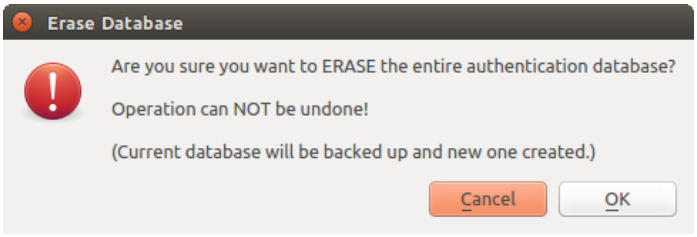


図 24.16: DB 消去検証メニュー

24.1.6 認証設定を使用する

典型的には、認証構成は、（例えば、WMS のような）ネットワークサービスのための設定ダイアログで選択されています。しかし、セレクタウィジェットは、サードパーティの PyQGIS または C ++プラグインのように、認証が必要などこにでも、または非コア機能中に、埋め込むことができます。

セレクタを使用しているとき、何も選択されていない場合、選択する構成がない場合、または以前に割り当てられた構成がもはやデータベースに見つからない場合は **認証なし** がポップアップメニューコントロールに表示されます。 *type* と *Id* フィールドは読み取り専用とそれぞれの認証方法の説明と構成の ID を提供しています。

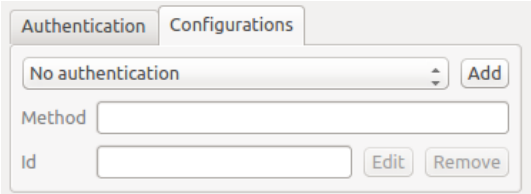


図 24.17: 認証なしの認証設定セレクタ

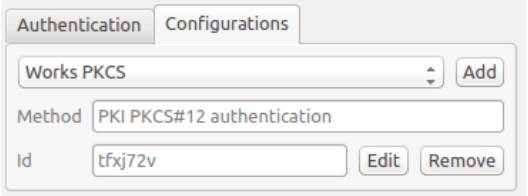


図 24.18: 構成が選択された認証設定セレクタ

24.1.7 Python バインディング

すべてのクラスとパブリック関数は SIP バインディングを持っています、 `QgsAuthCrypto` は除く。マスターパスワードのハッシュと認証データベース暗号化の管理はメインのアプリによって、そして Python を経由してではなく、処理されなければならないので。Python のアクセスに関しては **セキュリティの考慮事項** 参照。

24.2 ユーザー認証ワークフロー

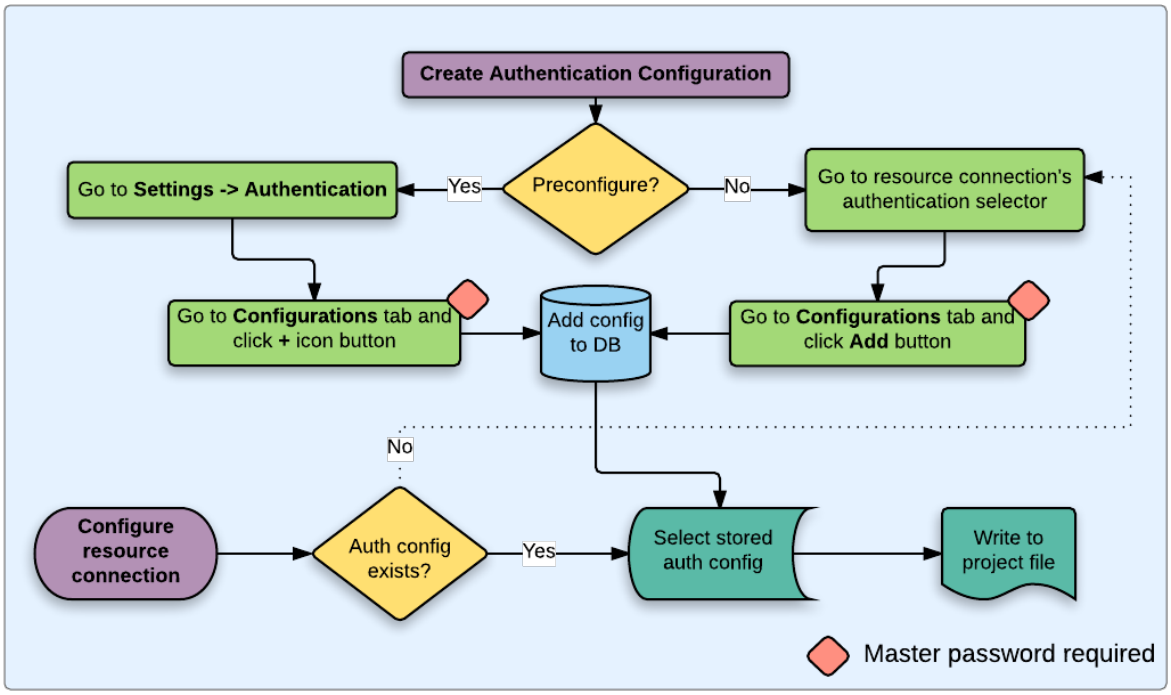


図 24.19: 一般的なユーザーのワークフロー

24.2.1 HTTP（S）認証

最も一般的なリソース接続の 1 つは、HTTP(S) を介して例えば Web マッピングサーバーで、認証方式のプラグインは多くの場合に接続のこれらのタイプのために機能します。方法プラグインは、HTTP リクエスト・オブジェクトにアクセスし、要求、ならびにそのヘッダの両方を操作できます。これにより、インター

ネットベースの認証の多くの形態が可能になります。標準のユーザー名/パスワードを使用して HTTP(S) 経由で接続するときは、認証方法は接続時に HTTP BASIC 認証を試みます。

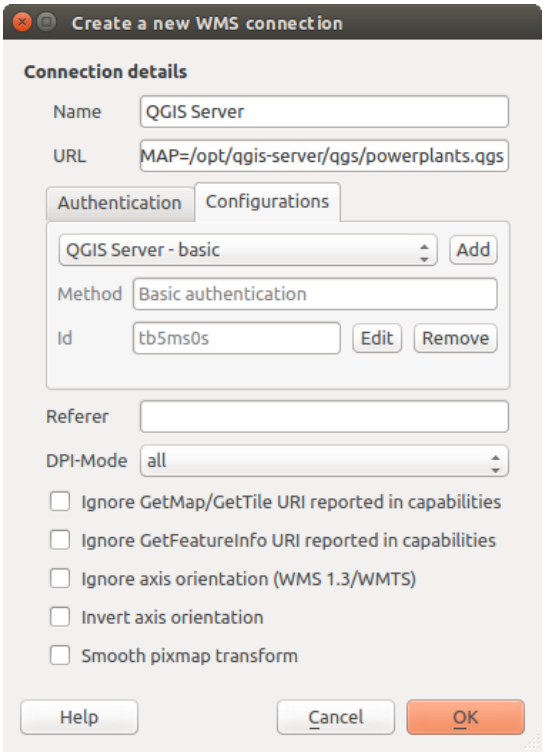


図 24.20: HTTP BASIC のための WMS 接続の設定

24.2.2 データベース認証

データベースリソースへの接続は一般的に `key=value` のペアとして保存され、認証設定を使用 **しない** 場合は、ユーザー名と（オプションで）パスワードが暴露されます。認証システムを使うよう設定した場合、`key=value` は認証情報の抽象化された表現になります（例 `authfg=81t21b9`。

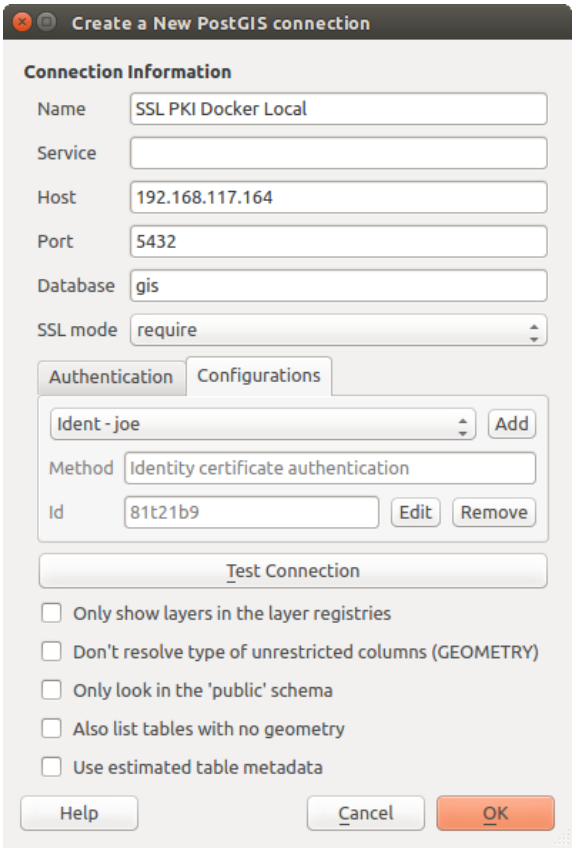


図 24.21: Postgres の PKI 付 SSL の接続を設定します

24.2.3 PKI 認証

認証システム内の PKI コンポーネントを設定するときは、データベースにコンポーネントをインポートしたり、ファイルシステムに保存されているコンポーネントファイルを参照するオプションがあります。このような成分が頻繁に変更、又はここでコンポーネントは、システム管理者によって置換される場合、後者は有用であり得ます。いずれの場合では、データベース内の秘密鍵にアクセスするために必要なすべてのパスフレーズを保存する必要があります。

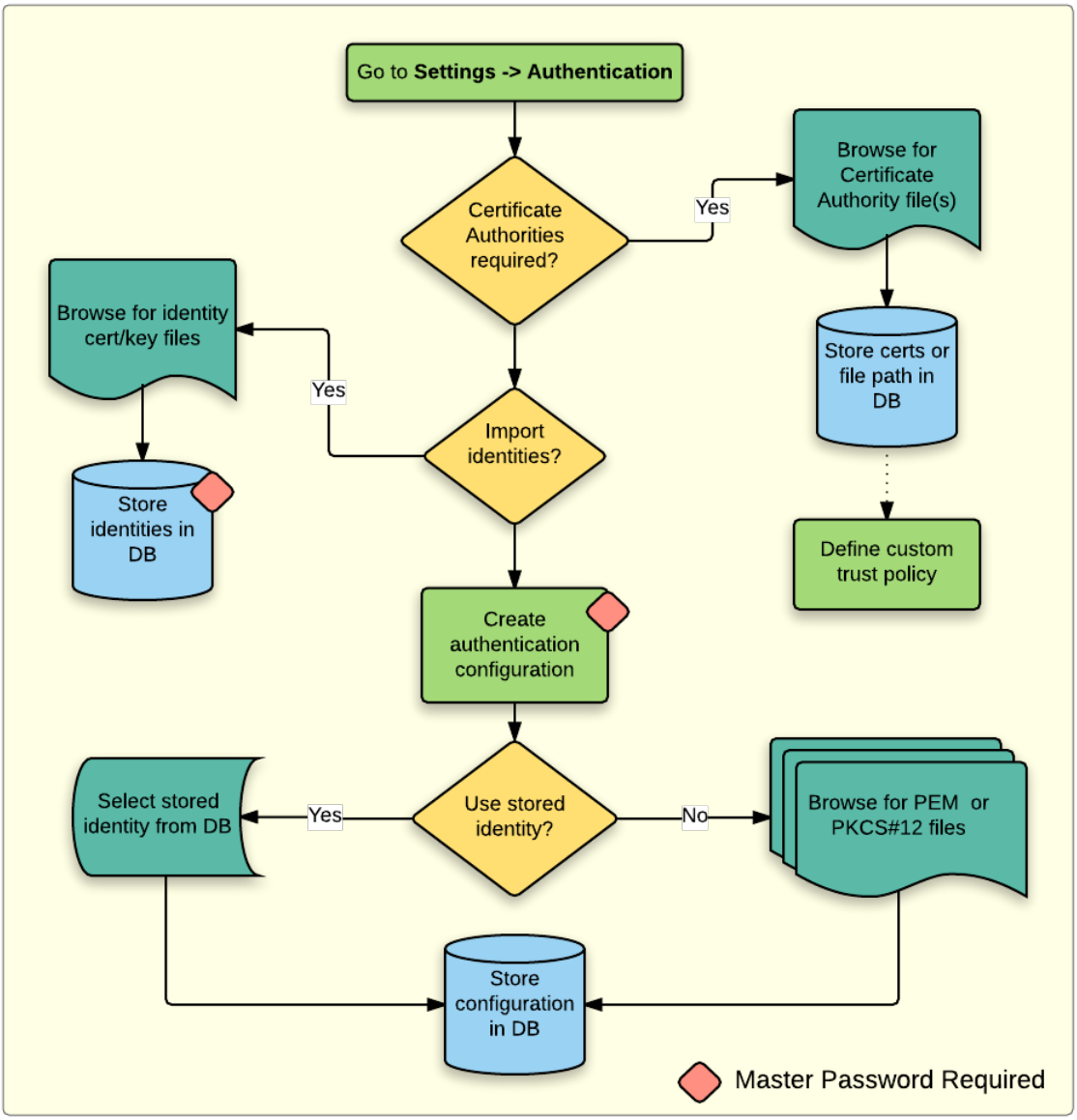


図 24.22: PKI 設定ワークフロー

すべての PKI コンポーネントは、**証明書マネージャ** 内の個別のエディタで管理することができます。このエディタは、QGIS の **オプション** ダイアログ（**設定** ▸ **オプション**）の **認証** タブで **証明書を管理** ボタンをクリックしてアクセスすることができます。

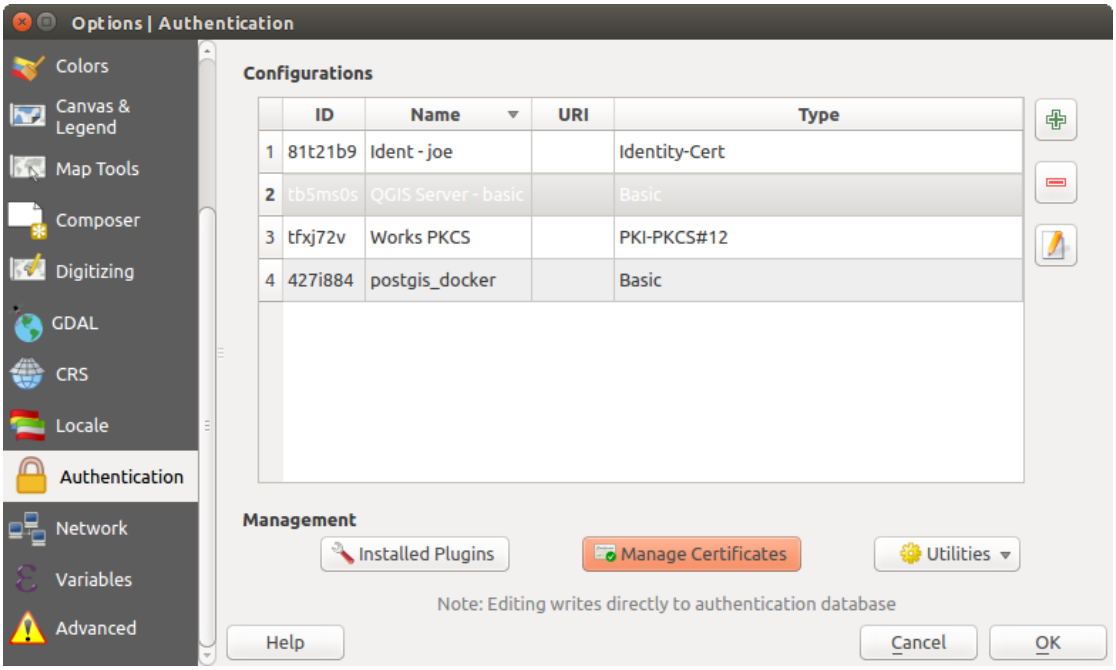


図 24.23: 証明書マネージャを開く

証明書マネージャ には、**Identities**、**サーバ**、**認証局** のエディタがあります。これらはそれぞれ独自のタブに含まれており、上記のワークフロー・チャートの中で遭遇する順番で以下に説明します。タブの順番は、ワークフローに慣れてきたら、頻繁にアクセスするエディタに相対するものです。

注釈: 認証システムの編集はすべて認証データベースに即座に書き込まれるため、変更を保存するために **オプション** ダイアログの *OK* ボタンをクリックする必要がありません。これは、オプションダイアログの他の設定とは異なります。

認証局

QGIS の **オプション** ダイアログの **認証** タブから、 **証明書マネージャ** 中の **認証局** タブから入手できる証明機関（CA）を管理できます。

上記ワークフローチャートで参照されるように、最初のステップは、インポートまたは CAS のファイルを参照することです。このステップはオプションで、商用証明書ベンダーからの証明書のように、PKI の信頼チェーンがオペレーティングシステム（OS）にインストールされているルート CA から発している場合は不要です。認証ルート CA が OS の信頼されたルート CA でない場合は、インポートするか、そのファイルシステムパスが参照されている必要があります。（不明な場合はシステム管理者に問い合わせてください。）

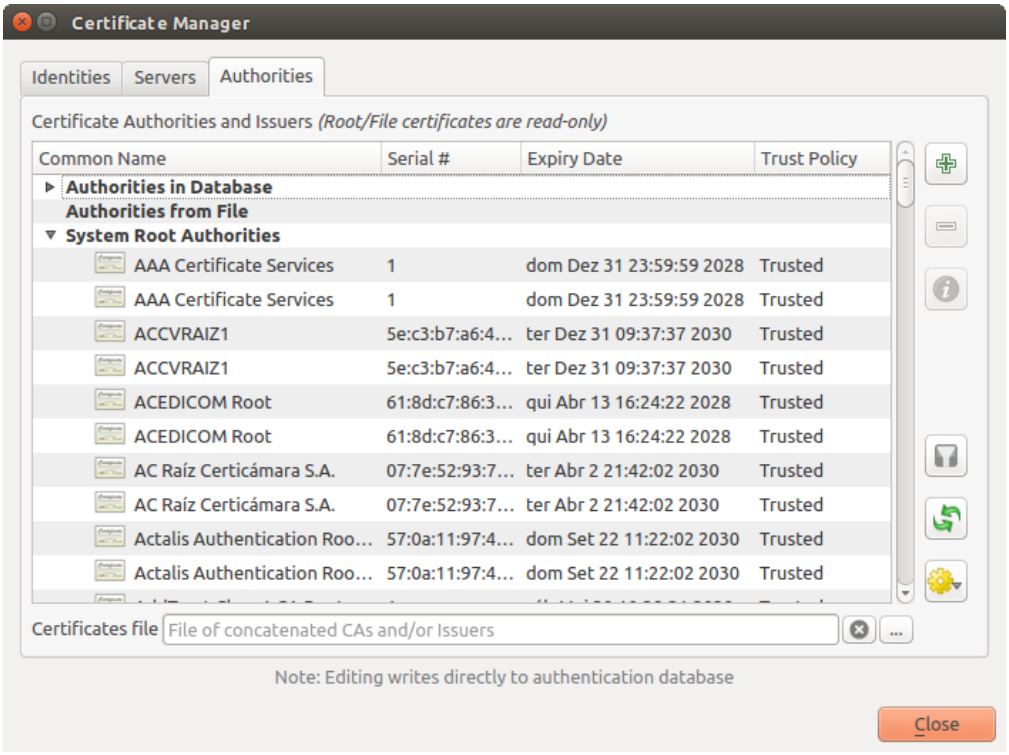


図 24.24: 認証局エディタ

デフォルトでは、OS のルート CA が利用可能ですが、その信頼設定は継承されません。特に OS のルート CA のポリシーが調整された場合は、証明書の信頼ポリシー設定を確認する必要があります。有効期限が切れた証明書は、特に信頼ポリシーを上書きしない限り、信頼されない設定になり、安全なサーバー接続に使用されません。証明書用の QGIS が検出可能な信頼チェーンを見るには、その証明書を選択し、 :sup:`証明書の情報を表示`をクリックします。

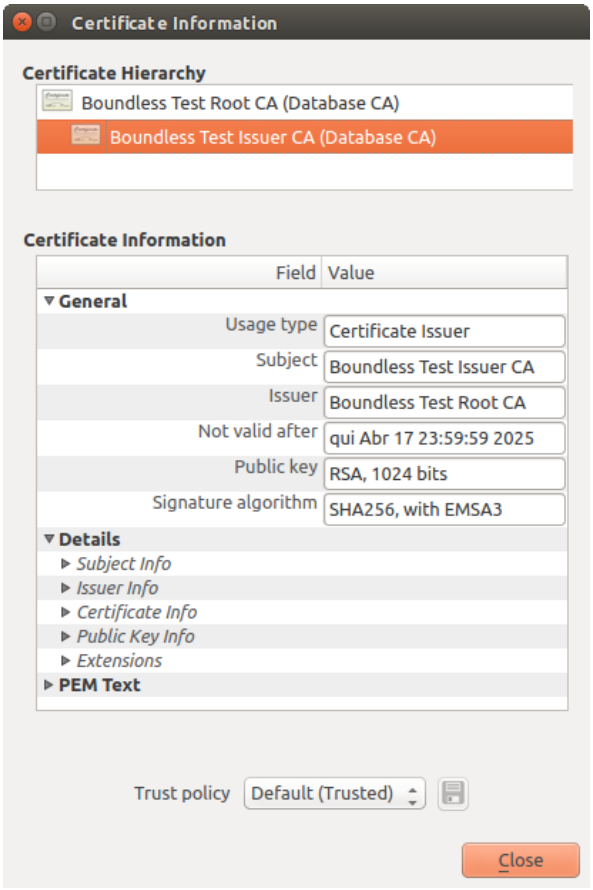


図 24.25: 証明書情報ダイアログ

チェーン内で選択した証明書の **信頼ポリシー** を編集することができます。選択した証明書に対する信頼ポリシーの変更は、選択した証明書 **ごと** に データベースに証明書信頼ポリシーの変更を保存します ボタンがクリックされない限り、データベースに保存されません。ダイアログを閉じても、ポリシーの変更は適用 **されません**。

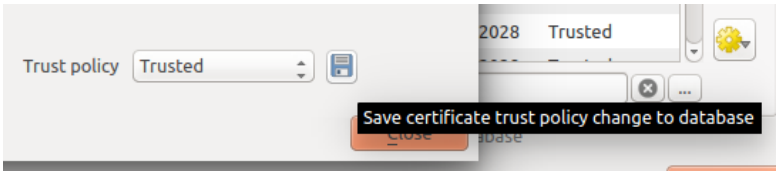


図 24.26: 信頼ポリシーの変更を保存する

セキュアな接続のために信頼されるフィルタの CA、両方の中間証明書とルート証明書を、確認またはクリックすることで、デフォルトの信頼ポリシーを変更できます を **オプション** ボタンを押します。

警告: デフォルトの信頼ポリシーを変更すると、安全な接続で問題が発生する可能性があります。

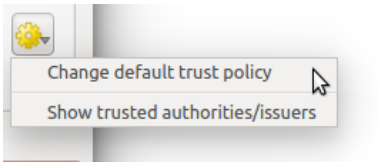


図 24.27: 当局のオプションメニュー

CA をインポートしたり、複数の CA が含まれているファイルからファイルシステムパスを保存したり、個々の CA をインポートすることができます。複数の CA チェーンの証明が含まれているファイルの標準 PEM 形式は、ファイルの一番下にルート証明書を持っており、すべてはその後、ファイルの先頭の方に、上記の子証明書に署名しました。

CA 証明書のインポート] ダイアログボックスが順序に関係なく、ファイル内のすべての CA 証明書を見つけ、また、(彼らの信頼ポリシーを上書きしたい場合には) 無効とみなされた証明書をインポートするオプションを提供していますでしょう。インポート時に信頼ポリシーを上書きするか、後でそれを **当局** エディタ内で行うことができます。

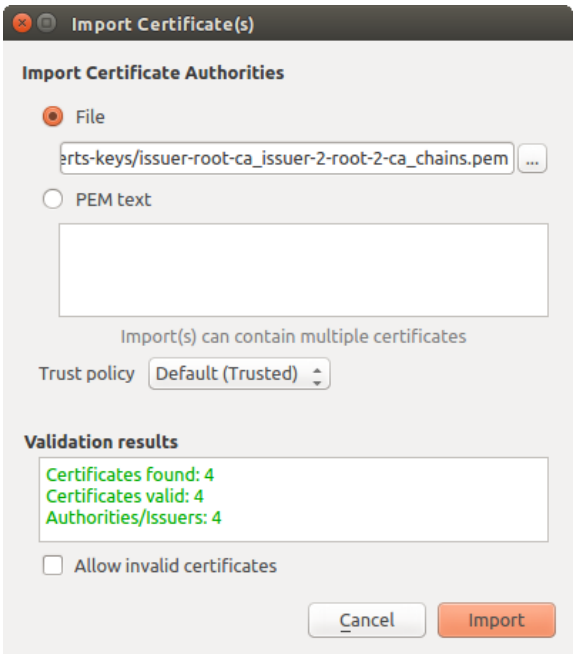


図 24.28: 証明書のインポート] ダイアログ

注釈: *PEM* の *text* フィールドに証明書情報を貼り付ける場合、暗号化された証明書がサポートされていないことに注意してください。

アイデンティティ

証明書マネージャ 認証 QGIS **オプション** ダイアログのタブ **アイデンティティ** タブでから入手できるクライアント ID のバンドルを管理できます。アイデンティティは、別々のファイルとして、または単一「バンドル」のファイルに結合のいずれか、PKI 対応サービスに対して、あなたを認証し、通常のクライアント証明書と秘密鍵で構成するものです。バンドルまたは秘密鍵は、多くの場合、パスフレーズで保護されています。

一度何らかの証明機関（CA）をインポートされると、必要に応じて認証データベースに任意のアイデンティティバンドルをインポートできます。アイデンティティを保存したくない場合は、個別の認証設定の中でそのコンポーネントファイルシステムパスを参照できます。

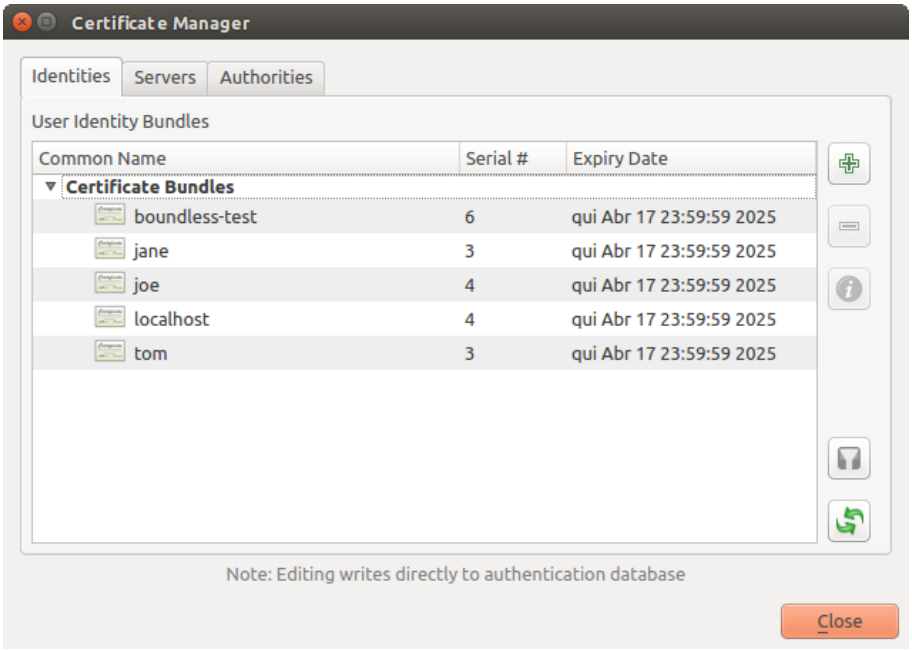


図 24.29: アイデンティティエディタ

アイデンティティ・バンドルをインポートする場合、パスフレーズで保護または非保護することができ、かつ信頼チェーンを形成する CA 証明書を含めることができます。トラストチェーンの認定は、ここではインポートされません。 **機関** タブ下でそれらは別途追加できます。

インポート時には、バンドルの証明書と秘密鍵は、鍵の保管は、QGIS のマスターパスワードを使用して暗号化し、データベースに保存されます。データベースから記憶された束のその後の使用は、マスターパスワードの入力を必要とします。

個人のアイデンティティは、PEM / DER（.PEM / .DER）と PKCS ＃ 12 からなるバンドル（.P12 / .PFX）コンポーネントがサポートされています。キーまたはバンドルがパスフレーズで保護されている場合は、パスワードをインポートする前にコンポーネントを検証する必要があります。バンドル内のクライアント証明書が無効である場合同様に、バンドルをインポートすることはできません（例えば、その効果的な日付はまだ開始されていないか、経過しています）。

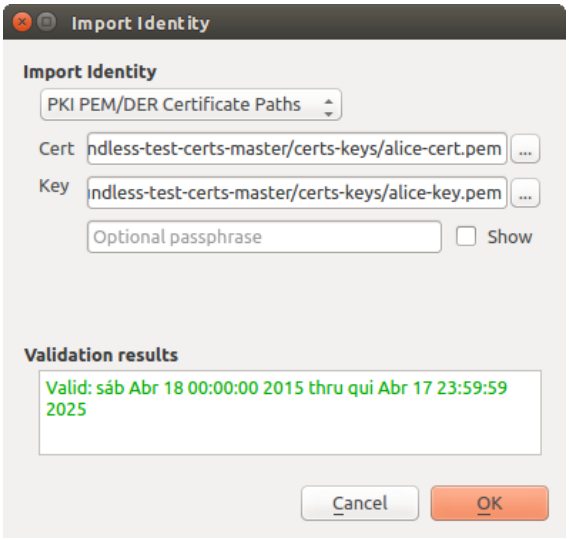


図 24.30: PEM / DER のアイデンティティのインポート

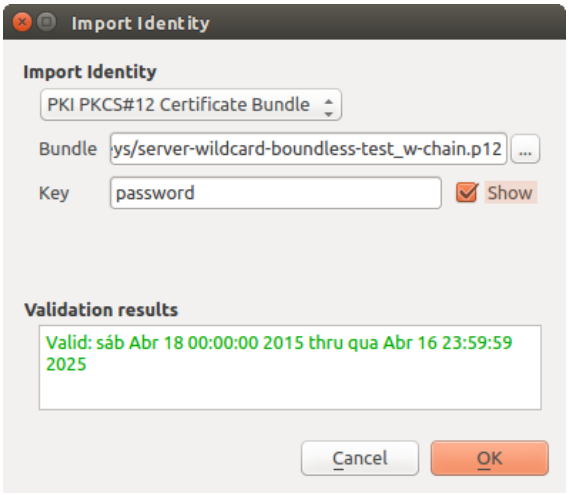


図 24.31: PKCS#12 identity import

24.2.4 不正レイヤの取り扱い

時折、プロジェクトファイルと一緒に保存された認証設定 ID が、おそらく現在の認証データベースがプロジェクトが最後に保存されたときと異なったり資格情報の不一致のせいで、もはや有効ではありません。このような場合には **不正レイヤ処理** ダイアログが QGIS の起動時に表示されます。

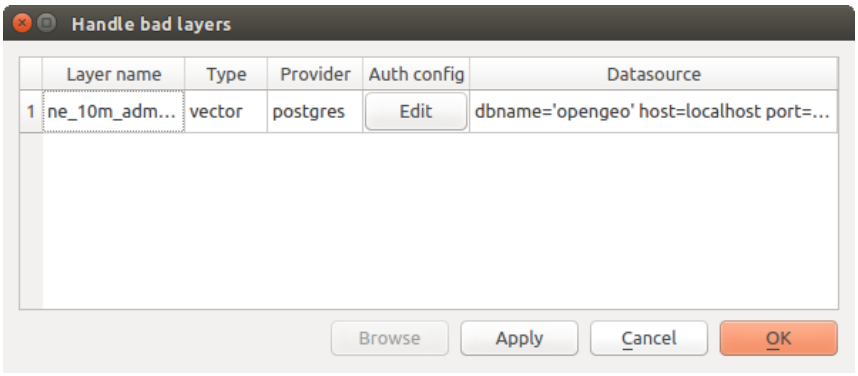


図 24.32: 認証で不正レイヤを処理する

データソースは、それに関連付けられた認証設定の ID を持つことが判明した場合、それを編集できます。そうすることで、自動的にテキストエディタでプロジェクトファイルを開くと、文字列の編集と同じように多くのデータソース文字列を編集します。

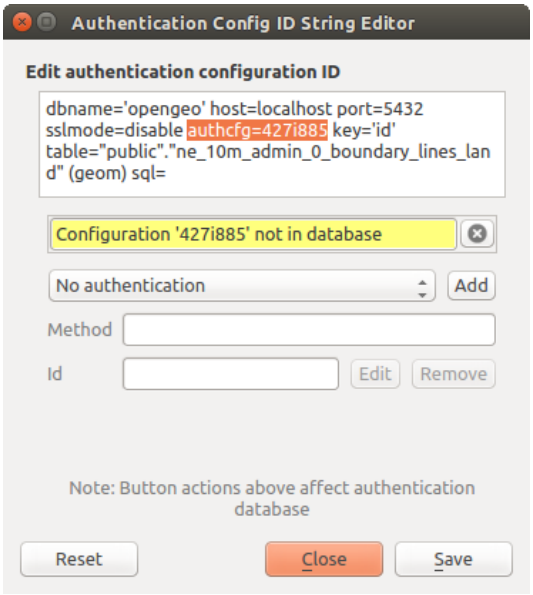


図 24.33: 不正レイヤの認証設定 ID を編集します

24.2.5 認証の設定の ID を変更する

リソースへのアクセスに関連する認証設定 ID の変更が必要な場合があります。これが便利な場合があります:

- **リソースの認証設定 ID が無効になっている**: これは、認証データベースを切り替えたときに、新しい設定をリソースに関連付けられた ID に合わせる必要がある場合に発生することがあります。
- **共有プロジェクトファイル**: 共有ファイルサーバーなどを介してユーザー間でプロジェクトを共有することを意図している場合、リソースに関連付けられた 7 文字（**a-z** や **0-9** を含む）を *predefine* が可能です。そして、個々のユーザーは、リソースの認証情報に特化した認証設定の ID を変更します。プロジェクトが開かれたとき、ID は認証データベースで見つかりますが、認証情報はユーザーごとに異なります。

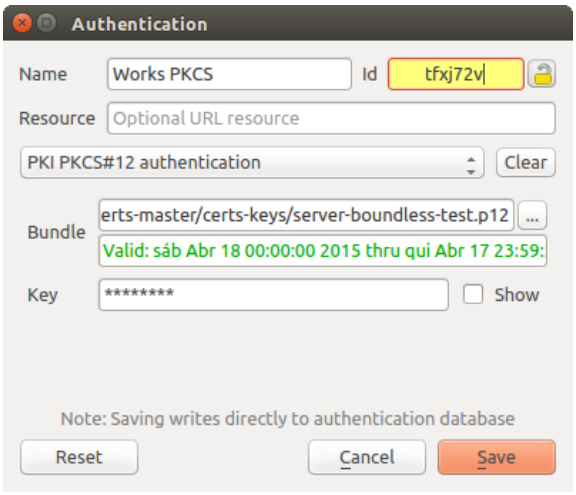


図 24.34: レイヤーの認証設定 ID（ロック解除された黄色のテキストフィールド）を変更します

警告: 認証設定 ID を変更することは高度な操作と考えられ、なぜそれが必要であるかの完全な知識を持ってのみ行われるべきです。ID を編集するのに先立って ID のテキストフィールドのロックを解除するためにクリックされる必要のあるロックボタンがあるのはこのためです。

24.2.6 QGIS サーバーのサポート

認証設定を持っているレイヤーがあるプロジェクトファイルを QGIS サーバー中の地図の基礎として使用する場合は、QGIS でそれらのリソースをロードするために必要な追加の設定手順がいくつかあります：

- 認証データベースが利用できるようにする必要があります
- 認証データベースのマスターパスワードが利用できるようにする必要があります

認証システムをインスタンス化する際、Server はアクティブな *user profile* に `qgis-auth.db` ファイルを作成または使用するか、`QGIS_AUTH_DB_DIR_PATH` 環境変数で定義したディレクトリを使用します。その場合、環境変数を使用して、サーバーのユーザーが読み取り/書き込み権限を持ち、ウェブアクセス可能なディレクトリ内にないディレクトリを定義してください。

サーバーにマスターパスワードを渡すには、サーバー・プロセスのユーザーが読み込み可能なファイルシステム上のパスにファイルの最初の行にそれを書くと `QGIS_AUTH_PASSWORD_FILE` 環境変数を使用して定義されました。サーバーのプロセスのユーザーによってのみ読めるようファイルを制限し、ウェブアクセス可能なディレクトリ内のファイルを保存しないように確認してください。

注釈: `QGIS_AUTH_PASSWORD_FILE` 変数は、アクセス後すぐにサーバー環境から削除されます。

24.2.7 SSL サーバーの例外

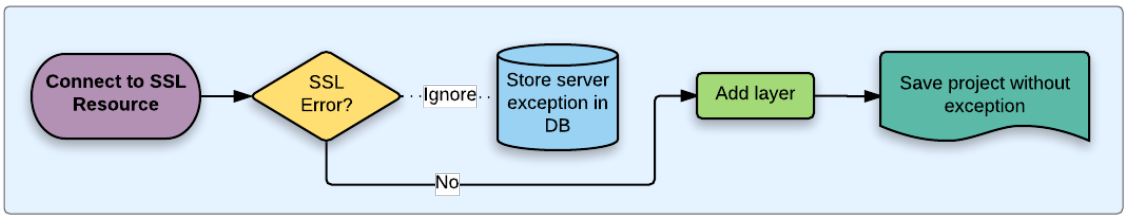


図 24.35: SSL サーバーの例外

QGIS の **オプション** ダイアログの **認証** セクション中の **サーバー** タブから、SSL サーバーの設定および例外を管理できます。

SSL サーバーへの接続時に時々、SSL「握手」や、サーバーの証明書でエラーがあります。これらのエラーを無視するか、例外として SSL サーバー構成を作成できます。これは、Web ブラウザを使用して SSL エラーをどのように上書きできるかに似ていますが、より細かく制御しています。

警告: サーバーとクライアントの間で全体の SSL の設定の完全な知識を持っていない限り、SSL のサーバー構成を作成しないでください。代わりに、サーバー管理者に問題を報告してください。

注釈: 一部の PKI のセットアップは、SSL サーバー証明書を検証するために使用するチェーンよりも、クライアントのアイデンティティを検証するために完全に異なる CA の信頼チェーンを使用します。このような状況では、接続するサーバー用に作成された任意の構成は、必ずしもあなたのクライアントのアイデンティティの検証の問題を解決しないだろう、自分のクライアント ID の発行者またはサーバー管理者だけが問題を解決できます。

ボタンをクリックすることにより、SSL サーバーの設定を事前に設定できます。SSL エラーが接続時に発生しますが **SSL エラー** ダイアログが表示されたときに代わりに、設定を追加できます（ここではエラーは一時的に無視されるか、またはデータベースに保存されて無視できます）：

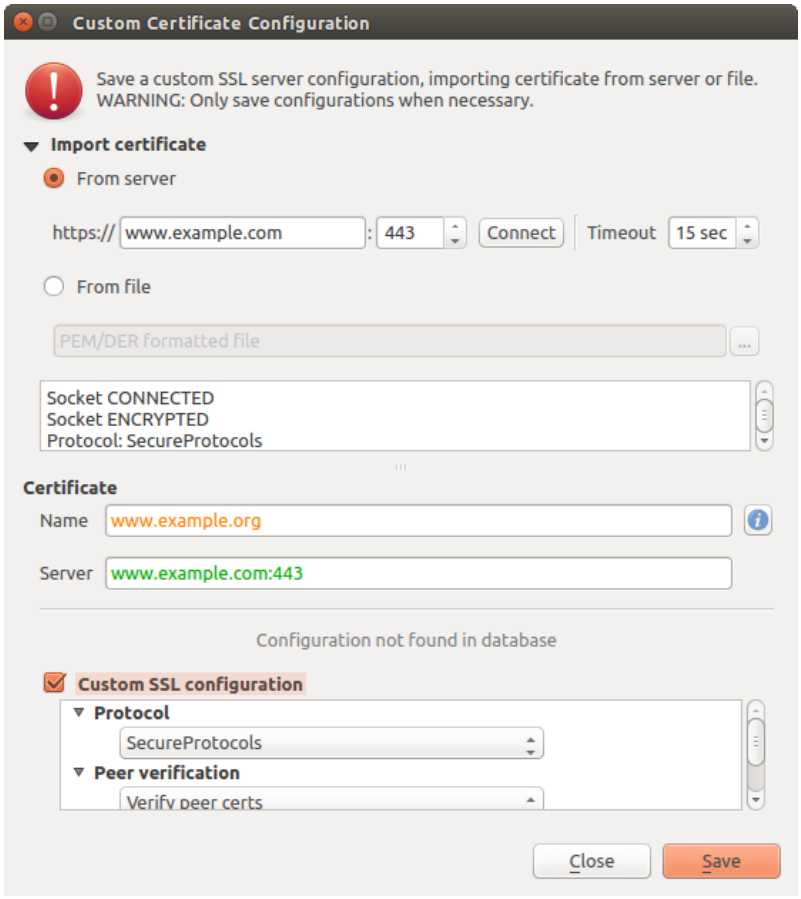


図 24.36: 手動で設定を追加

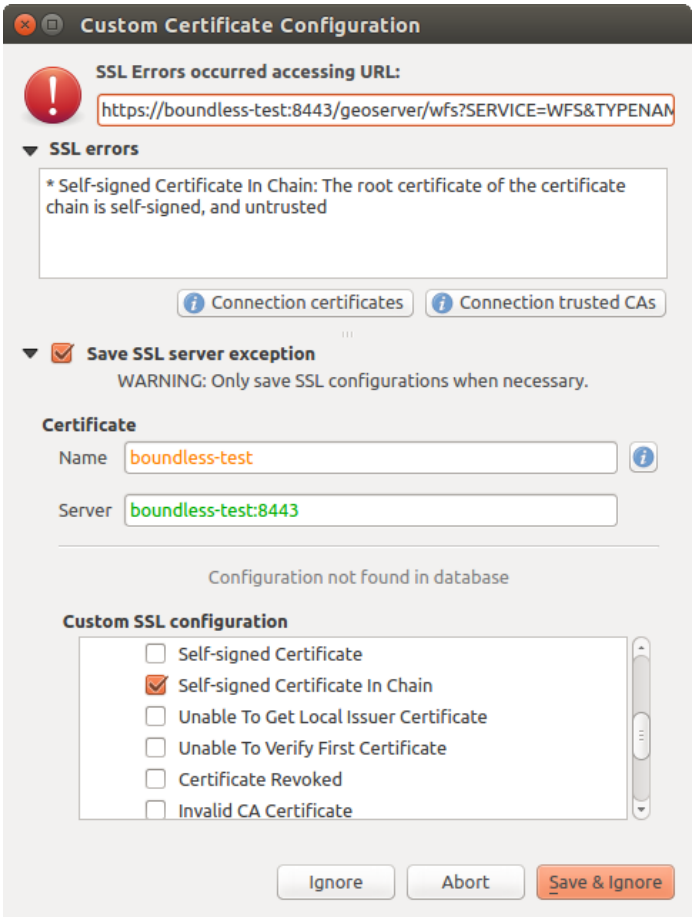


図 24.37: SSL エラー時の設定を追加

SSL の設定がデータベースに保存されれば、編集したり削除できます。

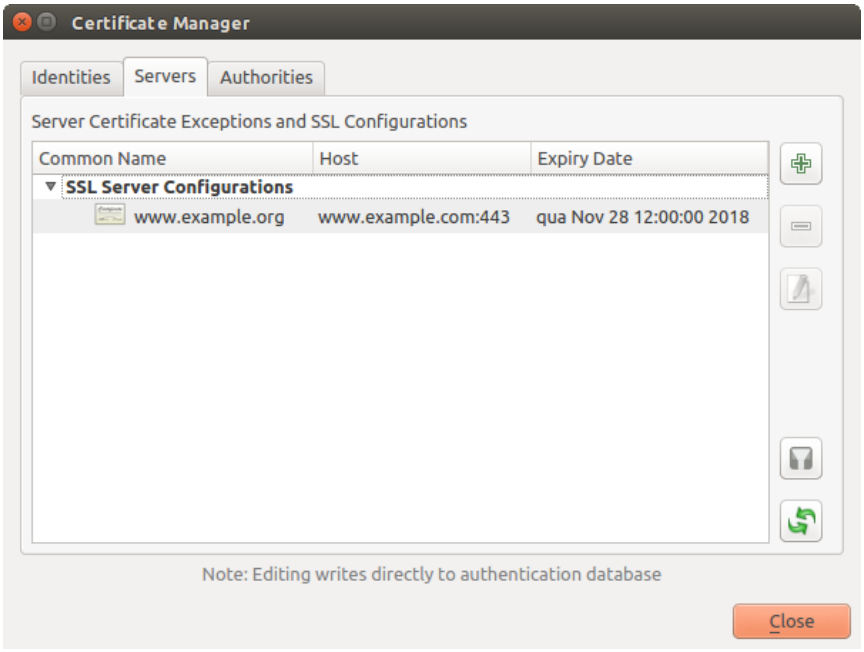


図 24.38: 既存の SSL 設定

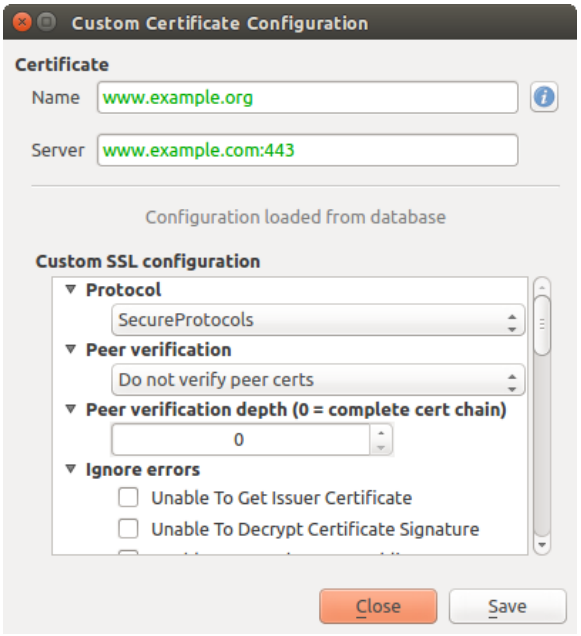


図 24.39: 既存の SSL 設定を編集します

SSL 設定を事前に行いたい場合で、インポートダイアログがサーバーの接続に対応していない場合は、以下のコードを実行することで、**Python Console** 経由で手動で接続をトリガーできます（*https://bugreports.qt-project.org* をサーバーの URL に置き換えてください）：

```
from qgis.PyQt.QtNetwork import QNetworkRequest
from qgis.PyQt.QtCore import QUrl
from qgis.core import QgsNetworkAccessManager

req = QNetworkRequest(QUrl('https://bugreports.qt-project.org'))
reply = QgsNetworkAccessManager.instance().get(req)
```

これにより、何かエラーが発生した場合は SSL エラーダイアログが開き、設定をデータベースに保存することを選択できます。

24.3 セキュリティの考慮事項

マスターパスワードが入力されると、API は、Firefox が機能する方法と同様、認証データベース中のアクセス認証 configs に開放されます。しかし、初期の実装では、PyQGIS アクセスに対しては壁が定義されていません。これは、認証証明書へのアクセスを獲得する悪意のある PyQGIS プラグインまたはスタンドアロンアプリケーションをユーザーがダウンロード／インストールする問題につながる可能性があります。

機能の最初のリリースのための手っ取り早い解決策は、単に認証システムのためにほとんどの PyQGIS バインディングを含めないことです。

別の簡単な、堅牢ではないけれども、修正は **設定** **オプション** **認証** でコンボボックスを追加することです（デフォルトは「never」）：

```
"Allow Python access to authentication system"
Choices: [ confirm once per session | always confirm | always allow | never]
```

このようなオプションの設定は、例えば、認証データベースの Python への非アクセスできる場所に保存し、マスターパスワードで暗号化する必要があろう。

- 別のオプションは、ユーザーが特にどのプラ　　グインを@認証システムへのアクセスを許可したか@追跡することかもしれません@
- 認証システムへのアクセスを許可@どのプラ　　グインが実際に呼び出しをしているかを推測するにはこつがいるかもしれませんが。
- プラグインを、おそらく自分の仮想環境で、サンドボックス設定すると、認証されている別のプラグインから認証コンフィグの「クロス・プラグイン」ハッキングを低減するでしょう。これは同様のクロス・プラグイン通信を制限する意味するかもしれませんが、多分サードパーティのプラグインとの間にのみ。
- もう一つの良い解決策は吟味されたプラグインの作者にコード署名証明書を発行することです。そのときはロード時にプラグインの証明書を検証します。必要であればユーザーは、直接既存の証明書の管理ダイアログを使用してプラグインに関連付けられた証明書を信頼されていないポリシーを設定できます。
- また、Python のよりセンシティブ認証システムのデータへのアクセス
- メインアプリの分野では、マスターパスワードと認証の設定負荷を維持しながら、認証の設定を持っているリソースで動作するようにプラグインを可能にする、許可されていない、と QGIS コアウィジェットの使用のみ、または認証システム統合を複製することがない可能性があります。

単に Python のと同じように削除するバインディング機能がないため、アクセスを制限することが困難になりますけれども、同じセキュリティ上の問題は、C ++のプラグインに適用されます。

24.3.1 制限事項

OpenSSL のに関連した紛らわしい ライセンスとエクスポート 問題が適用されます。Qt が SSL 証明書を使用して動作するためには、OpenSSL ライブラリにアクセスする必要があります。Qt がどうコンパイルされたかに応じて、デフォルトは（エクスポート制限を回避するために）実行時に動的に OpenSSL のライブラリにリンクすることです。

QCA は同様の戦術に従います、そこでは QCA にリンクすることは何の制限を招かない、なぜなら QCA-OSSL（OpenSSL の）プラグインが実行時にロードされるため。QCA-OSSL プラグインは、直接に OpenSSL LIBS にリンクされています。パッケージャは、彼らがプラグインを出荷する場合は、任意の OpenSSL の-リンクの制約が満たされていることを確認する必要なものだろう。多分。私は本当に知りません。私は弁護士ではありませんよ。

`qca-ossl` が実行時に見つからない場合、認証システムは安全に自身を無効にします。

第25章　GRASS GISの統合

GRASS統合は、GRASS GISデータベースと機能へのアクセスを提供します（**文献と *Web* 参照** 中のGRASS-PROJECTを参照）。統合は、プロバイダとプラグインの二つの部分から構成されています。プロバイダでは、GRASSラスタおよびベクタレイヤを閲覧、管理、視覚化できます。プラグインを使用すると、新しいGRASS locationとmapsetを作成し、GRASS領域を変更し、ベクタレイヤを作成・編集し、400の以上のGRASSモジュールでGRASS 2-Dと3-Dデータを分析できます。このセクションではプロバイダやプラグイン機能を導入し、GRASSデータを管理したりそれで作業するいくつかの例をあげます。

プロバイダはGRASSバージョン6および7をサポートし、プラグインはGRASS 6および7をサポートしています（QGIS 2.12以降）。QGISの配布には、GRASS 6またはGRASS 7、若しくは同時に両方のバージョン（バイナリは異なるファイル名を持ちます）のプロバイダ/プラグインを含んでいる場合があります。ただし、プロバイダ/プラグインのバージョンは実行時に1つだけ読み込めます。

25.1 デモデータセット

例として、QGISのアラスカデータセット（セクション **サンプルデータのダウンロード** を参照）を使用します。このデータセットには、3つのベクタレイヤと1つのラスタ標高マップを持った、小さなサンプルGRASSロケーション が含まれています。`grassdata` という新しいフォルダを作成し、https://qgis.org/downloads/data/ からQGIS 'Alaska' データセット `qgis_sample_data.zip` をダウンロードし、そのファイルを `grassdata` に展開します。

より多くのサンプルGRASSロケーション は、GRASSのウェブサイトhttps://grass.osgeo.org/download/data/で利用できます。

25.2 GRASSラスタおよびベクタレイヤを読み込む

プロバイダがQGISに読み込まれている場合、GRASS  アイコンの付いたロケーションアイテムが、GRASSロケーションを含む各フォルダアイテムの下のブラウザツリーに追加されます。フォルダ `grassdata` に移動し、ロケーション `alaska` と マップセット `demo` を展開します。

他のレイヤーと同様に、ブラウザからレイヤー項目をダブルクリックするか、マップキャンバスや凡例にドラッグ＆ドロップするかによって、任意のGRASSラスタレイヤとベクタレイヤを読み込めます。

Tip: GRASSデータの読み込み

GRASSロケーションアイテムが表示されない場合は、GRASSベクタプロバイダが読み込まれているかどうか、 **ヘルプ**　*QGIS* **について**　**プロバイダ** を検証してください。

25.3 ドラッグ＆ドロップで GRASS ロケーションへデータをインポートする

このセクションでは、GRASS mapset にラスタとベクタデータをインポートする方法を例示します。

1. QGIS のブラウザで、データをインポートしたいマップセットに移動します。
2. QGIS のブラウザで、GRASS にインポートするレイヤを探します。ソースデータがツリー内のマップセットから離れすぎている場合は、ブラウザの別のインスタンス（**ブラウザパネル（2）**）を開くことができることを覚えておいてください。
3. レイヤをドラッグし、インポート先のマップセットにドロップします。大きなレイヤのインポートには時間がかかることがあり、インポートが完了するまで、新たなレイヤ項目の前にアニメーションアイコン が表示されます。

ラスタデータが異なる CRS にある場合、それらは *Approximate*（高速）または *Exact*（正確な）変換を使用して再投影できます。元ラスタへのリンクが作成された場合（`r.external` を使用して）、元データは同じ CRS にあり、フォーマットが GDAL にはよく知られている、元データの CRS が使用されます。 *GRASS オプション* の **ブラウザ** タブでこれらのオプションを設定できます。

元ラスタがより多くのバンドを持っている場合、 **.<band number>** 接尾辞を持つ新しい GRASS 地図が各レイヤについて作成され、すべてのマップのグループ アイコンが作成されます。外部ラスタは別のアイコン を持ちます。

25.4 QGIS ブラウザで GRASS データを管理する

- マップをコピーする：GRASS マップは、同じロケーション内のマップセット間でドラッグアンドドロップを使ってコピーできます。
- マップを削除する：GRASS マップ上で右クリックしてコンテキストメニューから **削除** を選択します。
- マップの名前を変更する：GRASS マップ上で右クリックしてコンテキストメニューから **名前を変更** を選択します。

25.5 GRASS オプション

GRASS オプションは *GRASS* **オプション** ダイアログ中で設定できます。このダイアログはブラウザでロケーションまたはマップセットの項目を右クリックした後、 *GRASS* **オプション** を選択すると開きます。

25.6 GRASS プラグインを起動する

QGIS で GRASS の機能を使用するには、プラグインマネージャを使って GRASS プラグインを選択し、読み込む必要があります。これをするには、**プラグイン** **プラグインの管理とインストール...** メニューに行き、 *GRASS* を選択し、 :guilabel:`OK`をクリックします。

GRASS プラグインを起動すると、以下の主な機能が GRASS メニュー（ **プラグイン** *GRASS* ）で提供されています：

- Mapset を開く
- 新規 Mapset
- Mapset を閉じる
- GRASS ツールを開く
- 現在の GRASS 領域を表示
- GRASS オプション

25.7 GRASS mapset を開く

プラグインで GRASS ツールへのアクセスを取得するために GRASS mapset を開く必要があります（mapset が何も開かれていない場合はツールが無効になっています）。mapset はブラウザから開くことができます：mapset 項目を右クリックして、 *Mapset* **を開く** をコンテキストメニューから選択します。

25.8 GRASS[場所] と [地図セット]

GRASS のデータは GISDBASE と呼ばれるディレクトリに保存されます。このディレクトリは `grassdata` と呼ばれることが多く、QGIS で GRASS プラグインを使用する前に作成する必要があります。このディレクトリの中で、GRASS GIS データは `LOCATIONs` と呼ばれるサブディレクトリに格納されているプロジェクトごとに整理されています。各 `LOCATION` は座標系、地図投影法、地理的境界によって定義されます。各 `LOCATION` は複数の `MAPSETs` (`LOCATION` のサブディレクトリ) を持つことができ、それらはプロジェクトを異なるトピックやサブ領域に細分化したり、個々のチームメンバーのワークスペースとして使用されます (ref:*literature_and_web* の Neteler & Mitasova 2008 を参照)。GRASS モジュールでベクタやラスタのレイヤを分析するには、一般的に GRASS の `LOCATION` にインポートする必要があります。(これは厳密には正しくありません。GRASS モジュールの `r.external` と `v.external` を使えば、インポートせずに外部の GDAL サポートデータセットへの読み取り専用リンクを作成することができます。これは初心者が GRASS を使うときの一般的な方法ではないので、ここでは説明しません)。

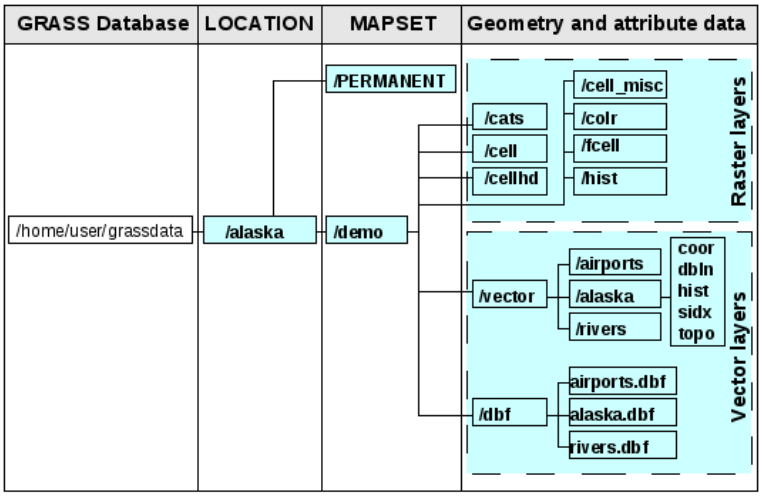


図 25.1: アラスカ LOCATION の GRASS データ

25.9 GRASS LOCATION へデータをインポートする

データを簡単にブラウザにドラッグ＆ドロップしてインポートできる方法を見つけるために、 **ドラッグ＆ドロップで *GRASS* ロケーションへデータをインポートする** セクションを参照してください。

このセクションでは、標準 GRASS モジュールを使用して、伝統的な方法で QGIS「アラスカ」データセットによって提供される「アラスカ」GRASS `LOCATION` に、ラスタおよびベクタデータをインポートする方法の例を示します。したがって、QGIS「アラスカ」データセットから土地被覆ラスター地図 `landcover.img` とベクター GML ファイル `lakes.gml` を使用します（ **サンプルデータのダウンロード** を参照）。

1. QGIS を起動し、GRASS プラグインがロードされていることを確認します。
2. GRASS ツールバーで、  Mapset を開く アイコンをクリックして、 *Mapset* ウィザードを起動します。
3. GRASS データベースとして、QGIS アラスカデータセットの `grassdata` フォルダを、 ロケーション として 'alaska'、 マップセット として 'demo' を選択し、 *OK* をクリックします。
4. ここで  GRASS ツールを開く アイコンをクリックします。GRASS ツールボックスダイアログが表示されます（セクション *GRASS* **ツールボックス** を参照）。
5. ラスター地図 `landcover.img` をインポートするには、 **モジュール** タブの `r.in.gdal` モジュールをクリックして下さい。この GRASS モジュールは GDAL がサポートしているラスタファイルを GRASS `LOCATION` にインポートします。 `r.in.gdal` モジュールダイアログが表示されます。
6. QGIS「アラスカ」データセット内の `raster` フォルダを参照して、ファイル `landcover.img` を選択します。
7. 出力ラスタマップ名として `landcover_grass` を定義し、 **実行** をクリックします。guilabel:*Output* `**タブに、現在実行中の** *GRASS* **コマンド** ``*r.in.gdal -o input=/path/to/landcover.img output=landcover_grass*`` が表示されます。
8. **成功しました** と表示されたら、 **出力を見る** をクリックします。これで `landcover_grass` ラスタレイヤが GRASS にインポートされ、QGIS キャンバスに表示されます。

9. ベクタ GML ファイル lakes.gml をインポートするには **モジュールツリー** タブのモジュール v.in.ogr をクリックします。この GRASS モジュールを使用すると、GRASS LOCATION に OGR サポートのベクターファイルをインポートできます。 v.in.ogr のモジュールダイアログが表示されます。

10. QGIS「アラスカ」データセット内の gml フォルダを参照し、OGR ファイルとして lakes.gml ファイルを選択します。

11. ベクタ出力名として lakes_grass を定義し、 **実行** をクリックします。この例では他のオプションを気にする必要はありません。**出力** タブには、現在実行中の GRASS コマンド v.in.ogr -o dsn=/path/to/lakes.gml output=lakes_grass が表示されます。

12. **成功しました** と表示されたら、**出力を見る** をクリックします。lakes_grass ベクタレイヤが GRASS にインポートされ、QGIS キャンバスに表示されます。

25.9.1 新しい GRASS[場所] を作成する

例として、単位としてフィートを使い、アルバース等積投影で投影された、サンプル GRASS [場所] アラスカ です。このサンプル GRASS [場所] アラスカ は、以下の GRASS 関連のセクションのすべての例と演習のために使用されます。コンピュータ上のデータセットをダウンロードしてインストールすると便利です（ **サンプルデータのダウンロード** を参照）。

1. QGIS を起動し、GRASS プラグインがロードされていることを確認します。

2. QGIS アラスカデータセット（ **サンプルデータのダウンロード** 参照）からの alaska.shp シェープファイル（セクション **ファイルからレイヤを読み込む** 参照）を可視化します。

3. GRASS ツールバーで、 地図セット アイコンをクリックして **新しい地図セット** ウィザードを起動してください。

4. 既存の GRASS データベース (GISDBASE) フォルダ grassdata を選択するか、コンピュータのファイルマネージャを使って新しい LOCATION 用のフォルダを作成します。そして **次へ** をクリックします。

5. このウィザードを使用して、既存の LOCATION の中に新しい MAPSET を作成したり（**新しい /地図セット/ を追加する** のセクションを参照）、完全に新しい LOCATION を作成することができます。 **新規ロケーションを作成** を選択します（参照 図 25.2）。

6. LOCATION の名前 - ここでは'alaska' とした - を入力し、:guilabel:`次へ`をクリックします。

7. 投影リストを有効にするラジオボタン **投影法** をクリックして投影を定義します。

8. 私たちはアルバース等積アラスカ（フィート）投影を使用しています。それは EPSG ID 2964 によって表されることを知っているので、検索ボックスに入力します。（注意：このプロセスを別の 場所 と投影法で繰り返したいが、EPSG ID を記憶していない場合は、ステータスバーの右下隅にある CRS Status アイコンをクリックしてください（ **投影法の利用方法** セクションを参照））。

9. **フィルタ** に 2964 と入力して投影法を選択して下さい.

10. **次へ** をクリックします。

11. デフォルト区域を定義するために、東西南北の LOCATION の境界を入力しなければなりません。ここで、ボタン **現在の *QGIS* の範囲を設定** をクリックすると、読み込まれたレイヤ alaska.shp の範囲が GRASS のデフォルト領域として適用されます。

12. **次へ** をクリックします。
13. 新しい :file: ‘ LOCATION` 内に MAPSET を定義する必要もあります（新しい LOCATION を作成するときはこれが必要です）。それには好きに名前を付けられます - ここでは'demo' を使いました。GRASS は PERMANENT と呼ばれる特別な MAPSET を自動的に作成します。それはプロジェクトのコアデータ、そのデフォルトの空間的範囲および座標系の定義を格納するように設計されています（ **文献と** *Web* **参照** 中の Neteler & Mitasova 2008 を参照）。
14. 要約が正しいことを確認し、**完了** をクリックします。
15. 新しい LOCATION 「アラスカ」 と 2 つの MAPSET 'demo' と'PERMANENT' が作られました. 現在開かれているワーキングセットはあなたが定義した 'demo' です。
16. GRASS ツールバーのそれまでは利用できなかったいくつかのツールが利用可能になっています.

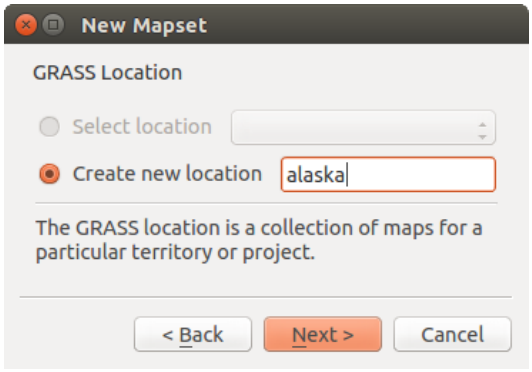


図 25.2: QGIS 内で新しい GRASS[場所] または新しい [地図セット] を作成する

手順が多いように感じたかもしれませんが、実際はそれほど悪くなく、LOCATION を素早く作成することができます。これで LOCATION 'alaska' はデータをインポートする準備が整いました（*GRASS LOCATION* **へデータをインポートする** セクションを参照）。また、QGIS 'Alaska' データセット **サンプルデータのダウンロード** に含まれるサンプルの GRASS LOCATION 'alaska' に既に存在するベクタデータとラスタデータを使って、セクション *GRASS* **ベクタデータモデル** に進むこともできます。

25.9.2 新しい [地図セット] を追加する

ユーザーは、自分で作成した GRASS MAPSET への書き込みアクセス権を持ちます。これは、自分の MAPSET へのアクセス以外にも、他のユーザーの MAPSET 中の地図を（そして他のユーザーはあなたのものを）読むことができることを意味しますが、変更したり削除できるの自分の MAPSET 中の地図だけです。

すべての MAPSET には WIND ファイルが含まれ, そこには現在の領域の座標値と選択されているラスタの解像度が格納されています (Neteler & Mitasova 2008 **文献と** *Web* **参照**, セクション *GRASS* **領域ツール** 参照).

1. QGIS を起動し、GRASS プラグインがロードされていることを確認します。
2. GRASS ツールバーで、  地図セット アイコンをクリックして **新しい地図セット** ウィザードを起動してください。
3. さらに「テスト」という MAPSET を追加するため、 LOCATION 「アラスカ」の GRASS データベース (GISDBASE) フォルダ grassdata を選択して下さい.

4. **次へ** をクリックします。
5. このウィザードを使用して、既存の LOCATION 内に新しい MAPSET を作成したり、全く新しい LOCATION を作成することができます。ラジオボタン **場所を選択** をクリックし（ 図 25.2 を参照）、 **次へ** をクリックします。
6. 新しい MAPSET に test という名前を入力します。ウィザードの下の方に、既存の MAPSET とそれに対応する所有者のリストが表示されます。
7. **次へ** をクリックし、要約が正しいことを確認し、**完了** をクリックします。

25.10 GRASS ベクタデータモデル

デジタイズに先立って GRASS ベクタデータモデル を理解することが重要です。一般的には、GRASS は、トポロジ的ベクタモデルを使用しています。これは、領域が閉じたポリゴンとしてではなく、一つ以上の境界で表現されることを意味します。二つの隣接する領域間の境界は一度だけデジタイズされ、それは、両方の領域によって共有されています。境界は隙間なく接続され閉じている必要があります。領域は、領域の **重心** によって識別（およびラベル付け）されます。

境界と重心のほかに、ベクタマップにはポイントとラインも含めることができます。これらのすべてのジオメトリ要素は、一つのベクタで混合できる一つの GRASS ベクタマップ内の異なるいわゆる「レイヤ」で表されます。だから GRASS においては、レイヤはベクタまたはラスタ地図ではなく、ベクタレイヤ内部の 1 つのレベルです。これは慎重に区別することが重要です。（ジオメトリ要素を混合することは可能ですが、それは異常なことであり、GRASS においてさえベクタネットワーク解析などの特別な場合に使用されるだけです。通常は、異なるジオメトリ要素は異なるレイヤに格納することをお勧めします。）

1 つのベクタデータセット内には複数の「レイヤー」を格納できます。例えば、田畑、森林、湖沼は、一つのベクタに格納できます。隣接する森林と湖沼には同じ境界を共有できますが、それらは別々の属性テーブルを持っています。境界に属性を付けることも可能です。例として、湖沼と森林の境界が道路なので、異なる属性テーブルを持つ場合があります。

地物の「レイヤ」は GRASS 内部で「レイヤ」によって示されます。「レイヤ」は（そのジオメトリが森か湖かなど）データセット内に複数のレイヤがあるかどうかを示す数値です。今のところ、それは数にだけできます。将来的には、GRASS は、ユーザーインターフェイス内のフィールドとして名前をサポートします。

属性は GRASS 場所 内部に dBase、SQLite3 として、または外部データベーステーブル 、例えば、PostgreSQL、MySQL、Oracle などに格納できます。

データベーステーブル中の属性は「カテゴリ」値を使用してジオメトリ要素とリンクされます。

「カテゴリ」（キー ID）はジオメトリプリミティブに付随した整数であり、それはデータベーステーブル内の 1 つのキー列へのリンクとして使用されます。

Tip: GRASS ベクタモデルについて調べる

GRASS ベクタモデルとその機能を学ぶ最良の方法は、ベクタモデルがより深く説明されている多くの GRASS チュートリアルの一つをダウンロードすることです。より詳しい情報、書籍、チュートリアルは https://grass.osgeo.org/learn/manuals/ をご覧ください。

25.11 新しい GRASS ベクタレイヤを作成する

新しい GRASS ベクタレイヤを作るには、ブラウザで地図セットコンテキストメニューから次の項目のいずれかを選択します：

- 点レイヤの新規作成
- 線レイヤの新規作成
- ポリゴンレイヤの新規作成

そして、ダイアログに名前を入力します。新しいベクタマップが作成され、レイヤはキャンバスに追加され、編集が開始されます。レイヤの種類を選択することは、ベクタマップにデジタイズできるジオメトリタイプを制限するものではありません。GRASS では、1 つのベクタマップですべてのジオメトリタイプ（ポイント、ライン、ポリゴン）をまとめることができます。QGIS はひとつのレイヤが特定のタイプを持っている必要があるので、種類はレイヤをキャンバスに追加するのに使われるだけです。

既存のベクタマップのコンテキストメニューから、上記項目のいずれかを選択し、既存のベクタマップにレイヤを追加することもできます。

GRASS はトポロジ的ベクタモデルを使用しているので、ひとつのレイヤでジオメトリタイプの全種類（ポイント、ラインおよびエリア）をまとめることができ、新しい GRASS ベクタを作成するときにジオメトリタイプを選択する必要はありません。これが QGIS でのシェープファイルの作成と異なっているのは、シェープファイルが単純地物ベクタモデルを使用しているためです（**新しいベクタレイヤを作成する** の項を参照）。

25.12 GRASS ベクタレイヤをデジタイズして編集する

GRASS ベクタレイヤは、標準の QGIS デジタイズツールを使用してデジタイズできます。しかし以下のためにいくつかの特殊性があり、これらは知っておくべきでしょう：

- GRASS トポロジモデル　対　QGIS 単純地物
- GRASS モデルの複雑さ
  - 単独のマップに複数のレイヤ
  - 単独のマップに複数のジオメトリタイプ
  - 複数のレイヤからの複数の地物によってジオメトリを共有します

その特殊性はこの後のセクションで説明します。

保存する、変更を破棄する、元に戻す、やり直す

> **警告:** 編集中に行われたすべての変更は、すぐにベクタマップと関連属性テーブルに書き込まれます。

変更は、各操作の後に書き込まれますが、編集を終了するときに、元に戻す/やり直す、すべての変更を破棄することができます。元に戻すまたは変更を破棄する場合、元の状態がベクタマップと属性テーブルに再び書き込まれます。

この動作には主な理由が 2 つあります：

- これは、ユーザーは自分がしていることがやりたいことであり、作業が突然中断されたとき（例えば停電）には、データが保存されている方が良い、という信念に由来する GRASS ベクタの性質です。
- トポロジ的なデータを効果的に編集するために必要なのは、トポロジ的に正しいかどうかの情報を視覚化することであり、そのような情報は、地図に変更が書き込まれた場合にのみ、GRASS のベクタ地図から得ることができます。

ツールバー

GRASS レイヤが編集されるときに「デジタイジングツールバー」にはいくつかの特定のツールがあります：

アイコン	ツール	目的
	点を追加	新しい点をデジタイズする
	線を追加	新しいラインをデジタイズする
	境界線を追加	新しい境界線をデジタイズする
	重心を追加	新しい重心をデジタイズする (ラベルのあるエリア)
	閉じた境界線を追加	新しい閉境界をデジタイズする

表 GRASS デジタイジング : GRASS デジタイジングツール

Tip: GRASS でポリゴンをデジタイズする

GRASS でポリゴンを作成する場合は、最初にポリゴンの境界線をデジタイズします。その後、閉じた境界線に重心（ラベルポイント）を追加します。この理由は、トポロジ的ベクタモデルはポリゴンの属性情報を、境界に対してではなく、常に重心に対してリンクしているためです。

カテゴリ

しばしば cat と呼ばれるカテゴリは、ID の一種です。その名前は GRASS ベクターのみが単独で「カテゴリ」属性持っていた時代から来ています。カテゴリは、ジオメトリと属性の間のリンクとして使用されています。単一のジオメトリは、複数のカテゴリを持ち、したがって異なるレイヤに複数の地物を表すことができます。現在は QGIS 編集ツールを使ってレイヤごとに 1 つのカテゴリだけを割り当てることができます。新しい地物は、境界線を除き、自動的に新しいユニークなカテゴリを割り当てます。境界線は通常は領域を形成し、線形の地物を表すものではありません。しかし、たとえば異なるレイヤで、後から境界線の属性を定義することができます。

新しいカテゴリは、常に現在編集中のレイヤにのみ作成されます。

QGIS 編集を使ってジオメトリに複数のカテゴリを割り当てることはできません。そのようなデータは複数の地物として適切に表され、個々の地物は、異なるレイヤからであっても、削除できます。

属性

現在編集中のレイヤの属性のみが変更できます。ベクタマップにより多くのレイヤが含まれている場合、他のレイヤの地物は、その属性が編集できないことを警告するため、すべての属性が '<not editable (layer

#)>' に設定されます。その理由は、QGIS がレイヤ毎に 1 つの固定されたフィールドセットしかサポートしないのに対して、他のレイヤが、異なるフィールドセットを持っことができ、通常は持っているからです。

ジオメトリプリミティブにカテゴリが割り当てられていない場合は、新しいユニークなカテゴリが自動的に割り当てられ、そのジオメトリの属性が変更されたとき、属性テーブルに新しいレコードが作成されます。

Tip: テーブル内の属性の一括更新を、たとえば「フィールド計算機」（ **フィールド計算機を使用する** ）を使って、行いたい場合で、更新したくないカテゴリを持たない地物（通常は境界線）がある場合、「高度なフィルタ」 `cat is not null` を設定することによりフィルタで除外できます。

編集スタイル

トポロジ的シンボルは、トポロジ的データを効果的に編集するために不可欠です。開始を編集する場合は、専門の「GRASS 編集」レンダラーをレイヤに自動的に設定され、編集が閉じているときに、元のレンダラーが復元されます。スタイルは、レイヤのプロパティ「スタイル」タブでカスタマイズできます。スタイルもプロジェクトファイルまたは任意の他のスタイルとして個別のファイルに保存できます。スタイルをカスタマイズする場合は名前を変更しないでください。名前は編集が再び開始されるときスタイルをリセットするために使用されます。

Tip: レイヤが編集されたときに、プロジェクトファイルを保存しないでください、レイヤは「編集スタイル」で格納されますが、レイヤが編集されていない場合には意味がありません。

スタイルは一時的にフィールド「topo_symbol」として属性テーブルに追加されたトポロジ情報に基づいています。フィールドは編集が閉じられたときに自動的に削除されます。

Tip: 属性テーブルから「topo_symbol」フィールドを削除しないでください。レンダラーはその列に基づいているため、地物が見えなくなるでしょう。

スナップ

エリアを形成するためには、接続された境界の頂点は **正確** に同じ座標を持っている必要があります。これは、キャンバスとベクタマップが同じ CRS を持っている場合にのみ、スナップツールを使用して達成できます。そうでない場合は、地図座標からキャンバスへのそしてその逆の正しい変換で、座標が表現誤差と CRS 変換のためにわずかに異なってくることがあります。

Tip: 編集するときはキャンバスについてもレイヤの CRS を使用してください。

制限

同時に同じベクタ内の複数のレイヤを同時編集することはサポートされていません。これは主に、単一のデータソースのスタックを元に戻す複数の処理が不可能であることに起因します。

Linux および Mac OS 上でだけ GRASS レイヤを一度に一つだけ編集できます。これは、ランダムな順序でデータベースドライバを閉じられないという GRASS のバグによるものです。これは、GRASS の開発者によって解決されてようとしています。

Tip: GRASS 編集権限

編集を行いたい場合は GRASS MAPSET の所有者になることが必要です。所有する以外の MAPSET に属するレイヤは、ファイルに書き込み権限を持っていても編集できません。

25.13 GRASS 領域ツール

GRASS の領域定義（空間的な作業ウィンドウを設定）は、ラスタレイヤを操作するために重要です。ベクタ分析はデフォルトで任意の定義された領域の定義に限定されていません。しかし、すべての新しく作成されたラスタは、元の範囲と解像度にかかわらず、現在定義されている GRASS 領域の空間的な範囲と解像度を持つことになります。現在の GRASS 領域は $LOCATION/$MAPSET/WIND ファイルに格納されており、それが北、南、東と西の境界、列と行の数、水平方向と垂直方向の空間解像度を定義します。


使用してスイッチを入れると QGIS キャンバスで GRASS 領域の可視化をオフすることが可能です
現在の GRASS の region
ボタンを表示します。

領域は「GRASS ツール」ドックウィジェットの「領域」タブで変更できます。新しい領域の境界と解像度を入力し、**適用** をクリックします。**キャンバス上のドラッグで領域を指定する** をクリックすると、QGIS キャンバス上でマウスを使って矩形をドラッグし、対話的に新しい領域を選択することができます。

GRASS モジュール g.region はラスタ分析のための適切な領域の広がりと解像度を定義するための多くのパラメータを提供します。 *GRASS* **ツールボックス** セクションで説明した、GRASS ツールボックスでこれらのパラメータを使用できます。

25.14 GRASS ツールボックス


GRASS ツールを開く
ボックスは、選択された GRASS LOCATION と MAPSET 内部のデータを操作するための GRASS モジュールの機能を提供します。GRASS ツールボックスを使用するには、書き込み権限を持っている LOCATION と MAPSET を開く必要があります（ MAPSET を作成した場合、通常は権限が付与されています)。分析中に作成された新しいラスターまたはベクターレイヤーは、現在選択されている LOCATION と MAPSET に書き込む必要があるため、これは必要です。

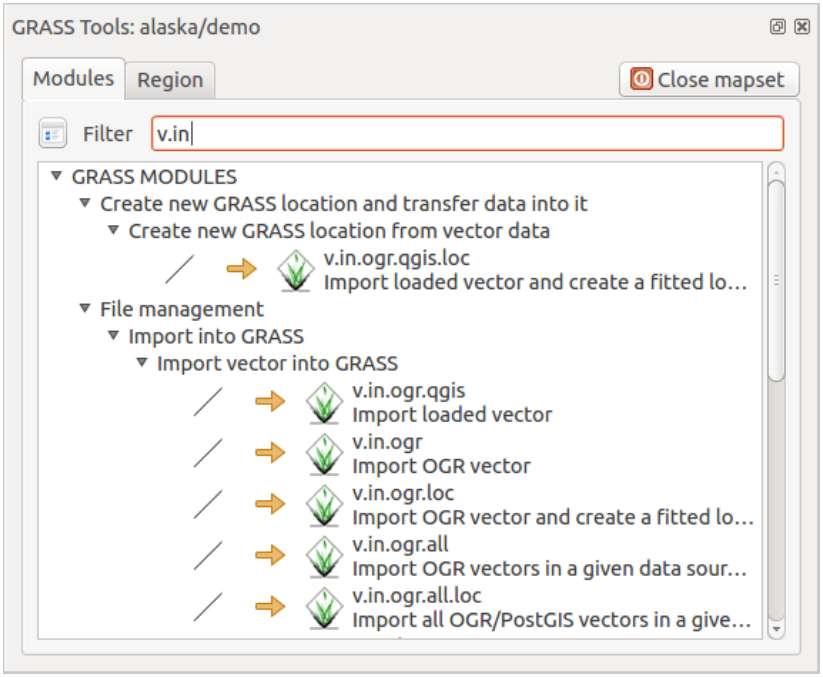


図 25.3: GRASS ツールボックスおよびモジュールツリー

25.14.1 GRASS モジュールを使用する

GRASS ツールボックスの GRASS シェルではほとんどすべての (300 以上)GRASS モジュールをコマンドラインインターフェイスで提供しています。もっとユーザーフレンドリな作業環境としてグラフィカルダイアログでは約 200 の GRASS モジュールと機能が提供されています。

QGIS バージョン 3.28 のグラフィカルツールボックスで利用可能な GRASS モジュールの完全な一覧は、GRASS wiki の https://grasswiki.osgeo.org/wiki/GRASS-QGIS_relevant_module_list で利用できます。

GRASS ツールボックスの内容をカスタマイズすることもできます. この手順は セクション *GRASS* **ツールボックスのカスタマイズ** に記述されています.

図 25.3 に示すように、テーマごとにグループ化された **モジュールツリー** や検索可能な **モジュールリスト** タブを使用して、適切な GRASS モジュールを探すことができます。

グラフィカルモジュールアイコンをクリックすると、新しいタブがツールボックス] ダイアログボックスに追加され、**オプション** 、**出力** と **マニュアル** の 3 つの新しいサブタブを提供します。

オプション

オプション タブでは、通常、QGIS キャンバスに可視化されたラスターまたはベクターレイヤーを選択し、モジュールを実行するために、さらにモジュール固有のパラメータを入力でき、簡略化モジュールダイアログを提供します。

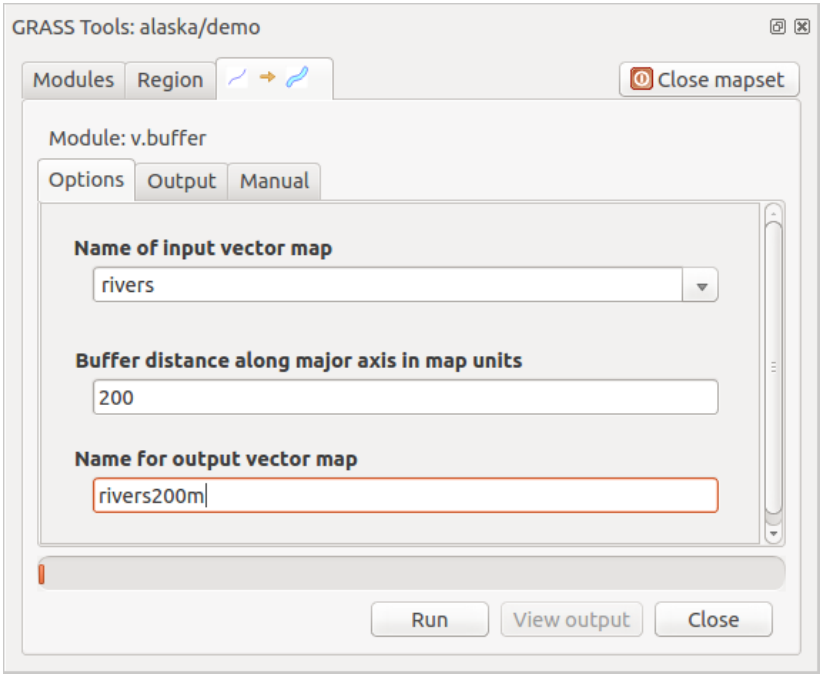


図 25.4: GRASS ツールボックスモジュールのオプション

提供されているモジュールパラメータは、ダイアログをシンプルに保つために完全でないことがよくあります。さらにモジュールパラメータとフラグを使いたい場合は、GRASS シェルを起動し、コマンドラインでモジュールを実行する必要があります。

QGIS 1.8 以降の新機能は、**オプション** タブの単純化されたモジュール] ダイアログの下にある **詳細オプションを表示します** ボタンです 。現時点では、使用例としてモジュール `v.in.ascii` にのみ追加されていますが、QGIS の将来のバージョンではそれはおそらくより多くの、またはすべての、GRASS ツールボックス内のモジュールの一部になります。これは GRASS シェルに切り替える必要なく、完全な GRASS モジュールオプションを使用できます。

出力

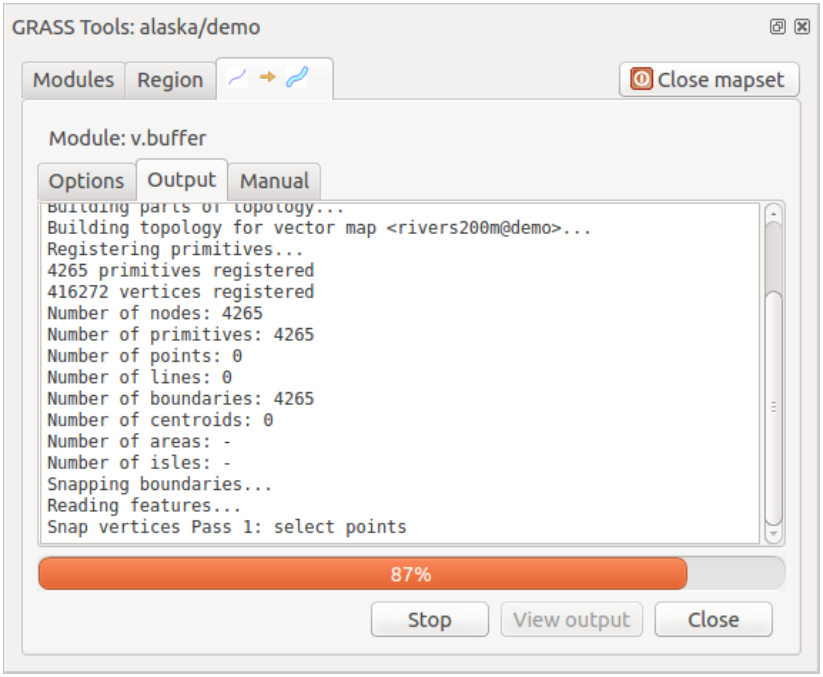


図 25.5: GRASS ツールボックスモジュール出力

出力 タブは、モジュールの出力状況に関する情報を提供します。**実行** ボタンをクリックすると、モジュールは **出力** タブに切り替わり、解析プロセスに関する情報が表示されます。すべてがうまくいくと、最後に 成功しました というメッセージが表示されます。

マニュアル

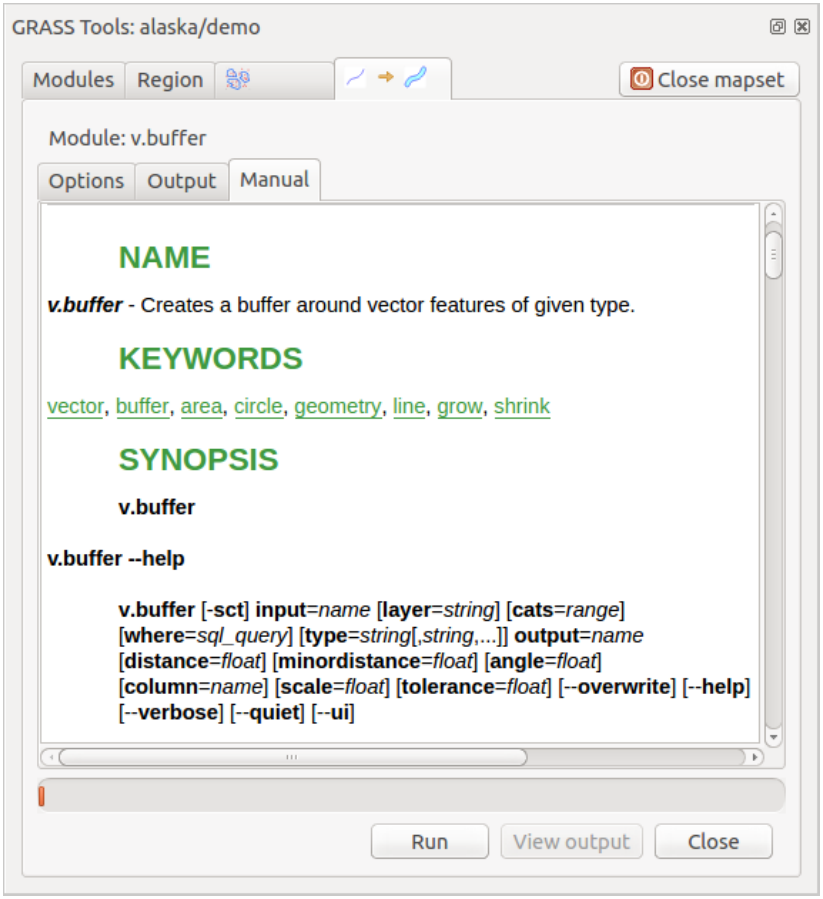


図 25.6: GRASS ツールボックスモジュールマニュアル

マニュアル タブは GRASS モジュールの HTML ヘルプページを示しています。それを使って他のモジュールパラメータとフラグを調べたり、モジュールの目的についてのより深い知識を得ることができます。各モジュールのマニュアルページの最後には、メインヘルプ索引 、 テーマ別索引 と 全索引 へのさらなるリンクがあります。これらのリンクは、モジュール `g.manual` と同じ情報を提供します。

Tip: 結果をすぐ表示

計算結果をすぐにマップキャンバスに表示したい場合、モジュールタブの一番下にある ' 出力を見る' ボタンを利用できます.

25.14.2 GRASS モジュールの例

以下の例はいくつかの GRASS モジュールの力を示すものです。

等高線を作成する

最初の例では、標高ラスタ（DEM）からベクタ等高線地図を作成します。 セクション *GRASS LOCATION* **へデータをインポートする** で説明したように、ここでアラスカ LOCATION を設定していることを想定しています。

- まず、  ボタンをクリックし、アラスカロケーションを選択してロケーションを開きます。
- ここで  ボタンでツールボックスを開きます。
- ツールカテゴリのリストで、*Raster*（**ラスタ**） *Surface management*（**表面管理**） *Generate vector contour lines*（**ベクタ等高線を生成**） をダブルクリックします。
- ツール **r.contour** をシングルクリックすると、上述したように、ツールダイアログが開きます（ *GRASS* **モジュールを使用する** を参照）。
- **入力ラスタマップ名** に gtopo30 を入力します。
- **等高線水準間の増加分** に  値 100 を入力します（これで等高線が 100 メートルの間隔で作成されます。）
- **出力するベクタマップ名** に ctour_100 と入力します。
- **実行** `をクリックして処理を開始します。**出力ウィンドウに** `**成功しました** `というメッセージが表示されるまでしばらく待ちます。そして **出力を見る** と **閉じる** をクリックします。

これは大きな領域ですので、表示されるまでに数分かかるでしょう。レンダリングを完了したら、レイヤのプロパティウィンドウを開き、 **ベクタプロパティダイアログ** のように、等高線が標高ラスタの上に明確に表示されるように線の色を変更できます。

次に、アラスカ州の中心部にある小さな、山岳地帯にズームインします。近くにズームインすると等高線が鋭い角を持っていることがわかります。GRASS ではそれらの全体の形状を維持しながら、少しベクタマップを変更する **v.generalize** ツールを提供しています。このツールは、異なる目的を持ついくつかの異なるアルゴリズムを使用しています。アルゴリズムの一部（すなわち、ダグラス・ペウカーおよび頂点削減）は、頂点の一部を除去することにより、ラインを簡素化します。得られたベクタは、より速く読み込まれます。非常に詳細なベクタを持っているが、非常に小縮尺の地図を作成しているので詳細が不要なとき、このプロセスが便利です。

Tip: 単純化ツール

QGIS にはちょうど GRASS **v.generalize** Douglas-Peuker アルゴリズムのように動作する **ベクタ ジオメトリツール ジオメトリを簡素化** ツールがあることに注意してください。

しかし、この例の目的は異なっています。 r.contour によって作成された等高線には鋭い角があって滑らかにする必要があります。 **v.generalize** アルゴリズムの中には、ちょうどそれをする Chaiken のもの（エルミートスプラインも）があります。これらのアルゴリズムはベクターに追加の頂点を **追加** し、読み込みがさらに遅くなることに注意してください。

- GRASS ツールボックスを開き、カテゴリ *Vector*（**ベクタ**） *Develop map*（**地図を展開**） *Generalization*（**簡素化**） をダブルクリックし、それから **v.generalize** モジュールをクリックしてそのオプションウィ

ンドウを開いてください。

- **入力ベクタの名前** として「ctour_100」ベクタが表示されていることを確認してください。
- アルゴリズムのリストから Chaiken's を選択します。他のオプションはデフォルトのままにしておき、最後の行までスクロールして **出力ベクタマップ名** フィールドに 'ctour_100_smooth' と入力し、 **実行** をクリックします。
- 処理にはしばらくかかります。出力ウィンドウに 成功しました が表示されたら、 **出力を見る** をクリックし、それから **閉じる** をクリックします。
- ラスタの背景に明確に表示するとともに元の等高線と対比するため、ベクタの色を変更することができます。新たな等高線は、元の全体的な形状に忠実でありながら、元々のものより角が滑らかになっていることが分かります。

図 25.7: ベクタマップを平滑化する GRASS モジュール v.generalize

Tip: その他に r.contour も使えます

上記の手順は、他の同等の状況で使用できます。降水量データのラスタマップを持っている場合は、例えば、同じ方法が等雨量（一定の降雨量）線のベクタマップを作成するために使用されます。

陰影 3D 効果の作成

いくつかの方法は、標高レイヤを表示し、地図に 3-D 効果を与えるために使用されます。上記のような等高線の使用は、多くの場合、地形図を生成するために選択される 1 つの一般的な方法です。3-D 効果を表示するための別の方法は、陰影起伏によるものです。陰影起伏効果は最初、空での太陽の位置をシミュレートし、各セルに反射率値を与え、その後、各セルの傾きと傾斜方向を計算することにより、DEM（標高）ラスタから作成されます。したがって、太陽に面した斜面が点灯取得します。（影で）太陽から離れて面する斜面を暗くしています。

- `gtopo30` 標高ラスタを読み込むことによって、この例を始めます。GRASS ツールボックスを起動し、ラスタカテゴリの下で、ダブルクリックして *Spatial analysis*（**空間分析**） *Terrain analysis*（**地形分析**） を開きます。

- そして **r.shaded.relief** をクリックしてモジュールを開きます。
- **方位角** を 1,00 270 から 315 に変更します 。
- 新しい陰影ラスタのために ``gtopo30_shade``と入力し、**実行** をクリックします。
- プロセスが完了すると陰影ラスタが地図に追加されます。これはグレースケールで表示されます。
- gtopo30 の陰影起伏と色の両方を一緒に表示するには、目次中の gtopo30 地図の下に陰影起伏図を移動し、次に *gtopo30* の **プロパティ** ウィンドウを開き、 **透明度** タブに切り替えて約 25 %の透明度レベルを設定`します。

これでカラーマップと透明度を設定した *gtopo30*` 標高がグレースケール陰影起伏図の **上に** 表示されるはずです。陰影の視覚効果を見るには、 gtopo30_shade マップをオフにして、それから再びオンにしてください。

GRASS シェルを使用する

QGIS の GRASS プラグインは、GRASS を初めて使い、すべてのモジュールやオプションに精通していないユーザ向けに設計されています。そのため、ツールボックスに表示されるモジュールの中には、オプションのすべてが表示されていないものや、まったく表示されていないものがあります。GRASS シェル（またはコンソール）は、ツールボックスのツリーに表示されない追加の GRASS モジュールへのアクセスをユーザーに提供し、また、ツールボックスにあるモジュールで最も簡単なデフォルトパラメーターを持つものに追加のオプションを提供します。この例では、上に出てきた **r.shaded.relief** モジュールの追加オプションの使い方を示します。

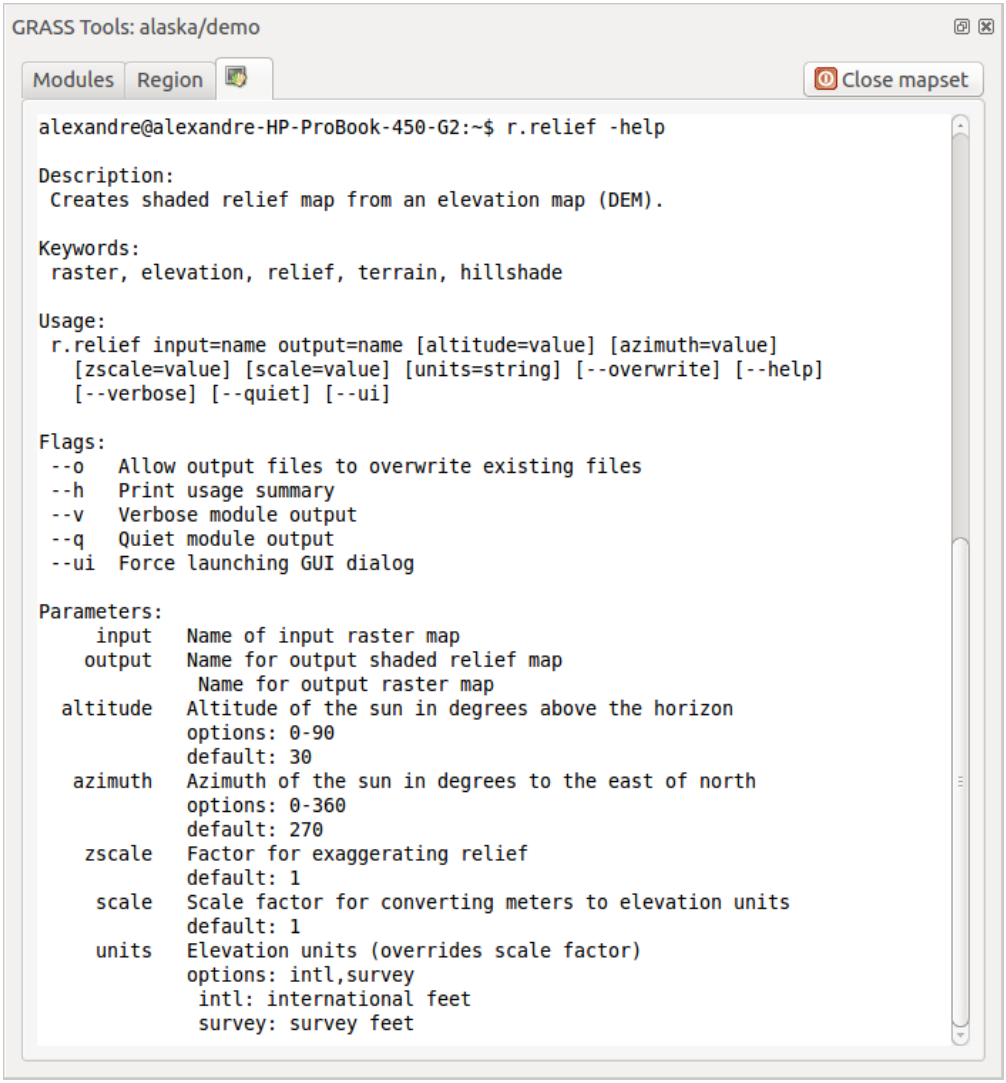


図 25.8: GRASS シェル、r.shaded.relief モジュール

モジュール **r.shaded.relief** は、陰影起伏効果がより顕著になるように、X-Y 座標単位に対して標高値に乗算するパラメータ、`zmult` を取ることができます。

- 上述したように `gtopo30` 標高ラスタを読み込んでから GRASS ツールボックスを起動し、GRASS シェルをクリックします。シェルウィンドウでコマンド `r.shaded.relief map=gtopo30 shade=gtopo30_shade2 azimuth=315 zmult=3` を入力し、 `Enter` を押します。
- プロセスが終了した後、 **ブラウズ** タブにシフトし、新しい `gtopo30_shade2` ラスタをダブルクリックして QGIS でそれを表示します。
- 上で説明したように、目次で陰影起伏ラスタを `gtopo30` ラスタの下に移動し、それから、着色された `gtopo30` レイヤの透明性を確認します。3-D 効果がより強く最初の陰影起伏図と比べて際立っていることがわかります。

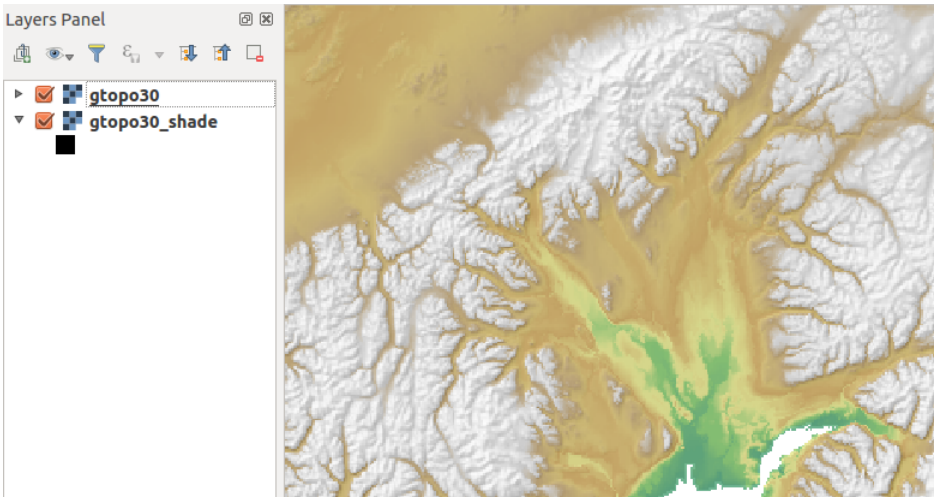


図 25.9: GRASS モジュール r.shaded.relief で作成した陰影を表示します

ベクタマップによるラスタ統計

次の例は GRASS モジュールがラスタデータを集計してベクタマップのそれぞれのポリゴンの列に統計値を追加するものです。

- 再びアラスカのデータを使って、 *GRASS LOCATION* **へデータをインポートする** を参照して、`shapefiles/trees.shp` ファイルを GRASS にインポートします。
- ここで、中間のステップが必要です：重心をインポートされた樹木図に追加して（境界と重心の両方を含む）完全な GRASS エリアベクタを作る必要があります。
- ツールボックスで *Vector*（**ベクタ**） *Manage features*（**地物を管理**）を選択して **v.centroids** モジュールを開きます。
- **出力するベクトルマップ名** として 'forest_areas' と入力してモジュールを実行して下さい.
- 次に、 `forest_areas` ベクタを読み込んで、森林の種類 - 落葉、常緑、混合 - を異なる色で表示します：レイヤの **プロパティ** ウィンドウの **シンボロジ** タブの中で、 **凡例タイプ** から 「ユニークな値」選び、 **分類フィールド** に「VEGDESC」を設定します。（ベクタセクションの **シンボロジプロパティ** 中のシンボロジタブの説明を参照してください。 ）
- 次に GRASS ツールボックスを再び開いて **ベクター** **ベクター更新** を他の地図で開いて下さい.
- **v.rast.stats** モジュールをクリックして `gtopo30` と `forest_areas` と入力して下さい.
- 追加で必要なパラメータは 1 つだけです： *column prefix* `elev` と入力し、 **実行** をクリックします。これは計算量が多く、長時間（おそらく 2 時間程度）実行される。
- 最後に、 `forest_areas` 属性テーブルを開き、各森林のポリゴンのために、いくつかの新しい列が、`elev_min` 、 `elev_max` 、 `elev_mean` などを含め、追加されていることを確認してください。

25.14.3 GRASS ツールボックスのカスタマイズ

ほぼすべての GRASS モジュールは、GRASS ツールボックスに追加できます。XML インターフェイスは、ツールボックス内のモジュールの外観とパラメーターを設定する非常に単純な XML ファイルを解析するために提供されます。

モジュール `v.buffer` （v.buffer.qgm）を生成するためのサンプル XML ファイルは次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE qgisgrassmodule SYSTEM "http://mrcc.com/qgisgrassmodule.dtd">

<qgisgrassmodule label="Vector buffer" module="v.buffer">
        <option key="input" typeoption="type" layeroption="layer" />
        <option key="buffer"/>
        <option key="output" />
</qgisgrassmodule>
```

モジュールを選択するとパーサーはこの定義を読み取り、ツールボックス内に新しいタブを作成します。新しいモジュールの追加、モジュールのグループの変更など、より詳しい説明は https://qgis.org/en/site/getinvolved/development/addinggrasstools.html にあります。

第26章　QGIS プロセシングフレームワーク

26.1 はじめに

この章では、QGIS におけるジオプロセシング環境で、ネイティブおよびサードパーティのアルゴリズムを呼びだして使うことのできる、QGIS プロセシングフレームワークについて紹介します。これを使うと空間分析作業をより生産的に簡単に行うことができます。

プロセシングは **コアプラグイン** として最初からインストールされていますが、有効化する必要があります。

1. メニューで **プラグイン**　**プラグインの管理とインストール**... を選択します。
2. 左端のコラムで **インストール済み** タブをクリックします。
3. 表示されている項目の中から  *Processing* にチェックを入れます。
4. ダイアログを閉じます。

   最上端のメニューバーで **プロセシング** メニューが有効になります。ここからプロセシングフレームワークの主要コンポーネントにアクセスできます。

以降のセクションではこのフレームワークのグラフィカルなツールをどのように使うのかを説明してそれぞれのツールを最大限に活用します。

このフレームワークの GUI には 4 つの基本的な要素があり、多様な目的のアルゴリズムを実行するために使われます。どのツールを選ぶかは、行われる分析の種類と、それぞれのユーザとプロジェクトの特性に依存します。すべてのツールは、バッチ処理インターフェイスを除き、**プロセシング** メニュー項目からアクセスすることができます（バッチ処理は、後で見るように、ツールボックスかアルゴリズム実行ダイアログから呼ばれます）。メニューにはその他の選択肢も表示されると思いますが、それらはアルゴリズムの実行には使われませんので、この章の後の方で説明します。

- **ツールボックス**： この GUI の主要要素です。単一のアルゴリズムか、もしくはそのアルゴリズムに基づいたバッチ処理を走らせるために使われます。

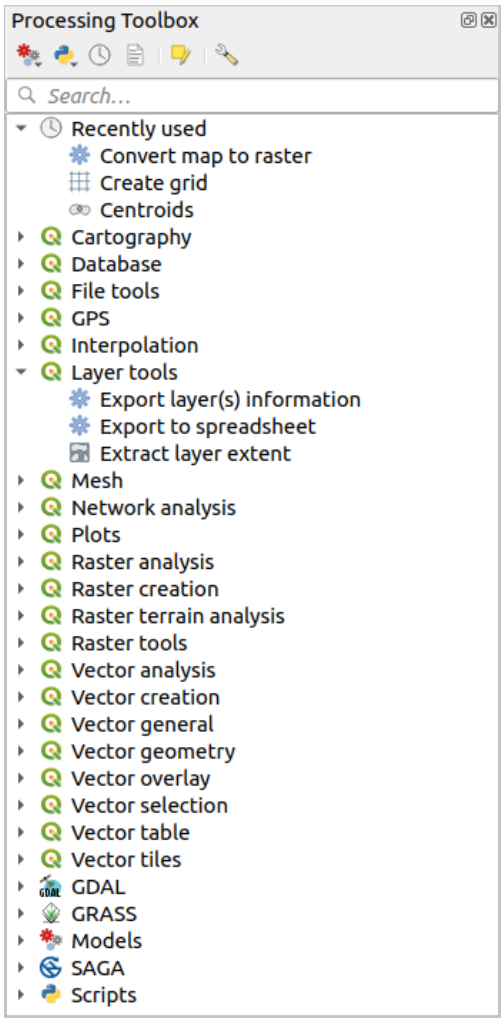


図 26.1: プロセシングツールボックス

- **モデルデザイナー**： 複数のアルゴリズムをモデラーを用いてグラフィカルに組み合わせることにより、ワークフローを定義することができます。これによって複数のサブプロセスを伴って実行されるひとつのプロセスを作成します。

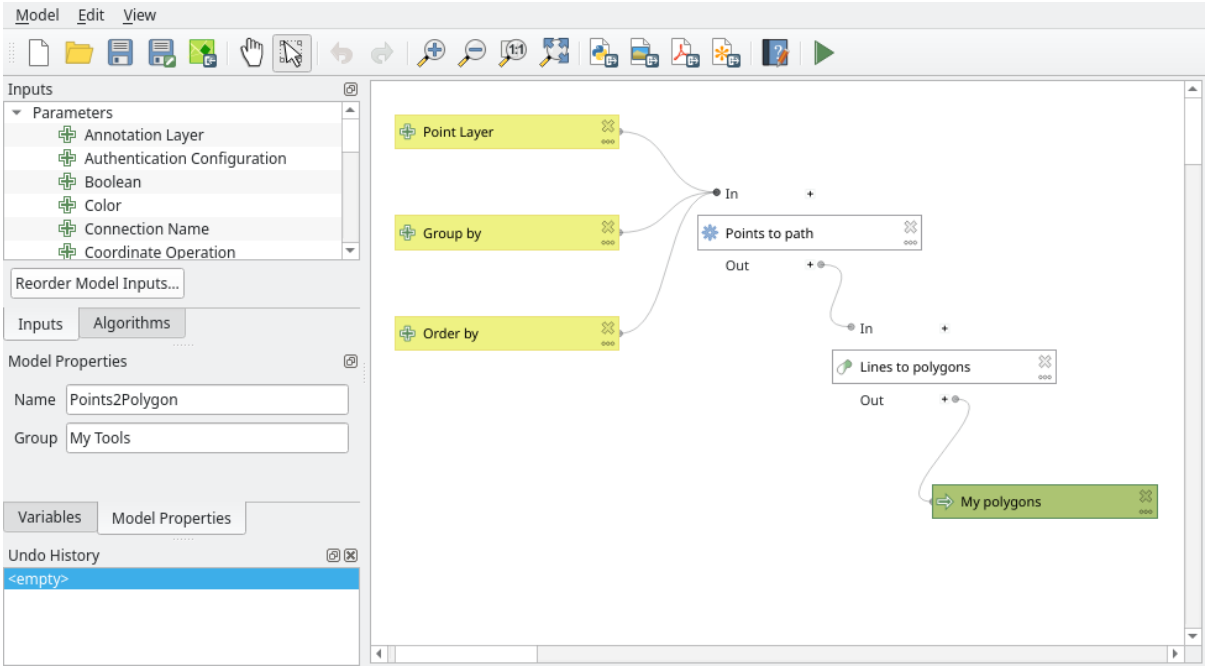


図 26.2: プロセシングモデラー

- **履歴** マネージャ： 先の 2 つの要素で行われた操作はすべて履歴ファイルに格納され、履歴マネージャを使って後から簡単に繰り返すことができます。

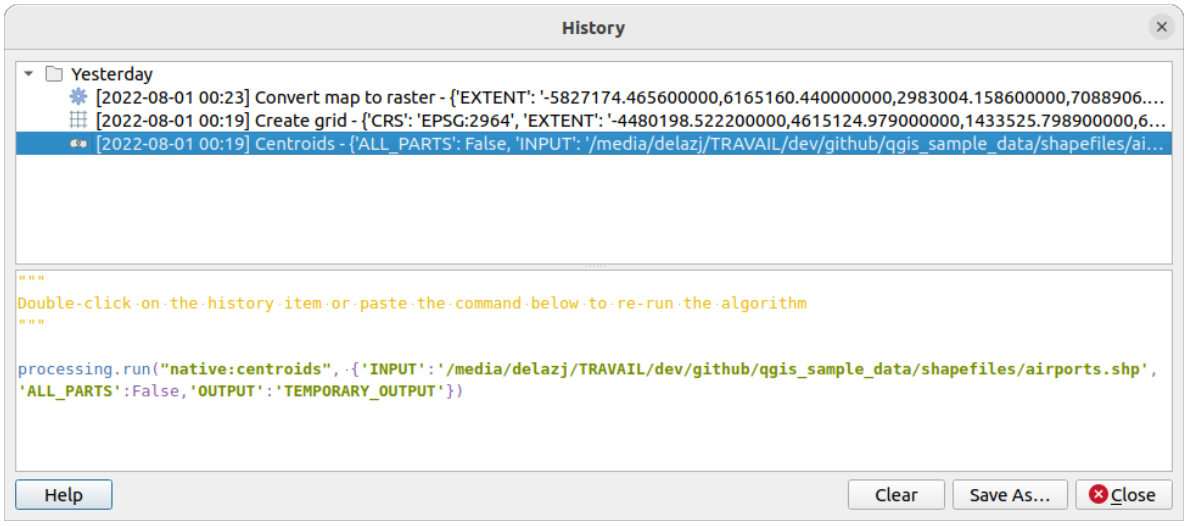


図 26.3: プロセシング履歴

- **バッチ処理** インターフェイス： このインターフェイスでバッチ処理を行うことにより、複数のデータセットに対する単一のアルゴリズムの実行を自動化することができます。

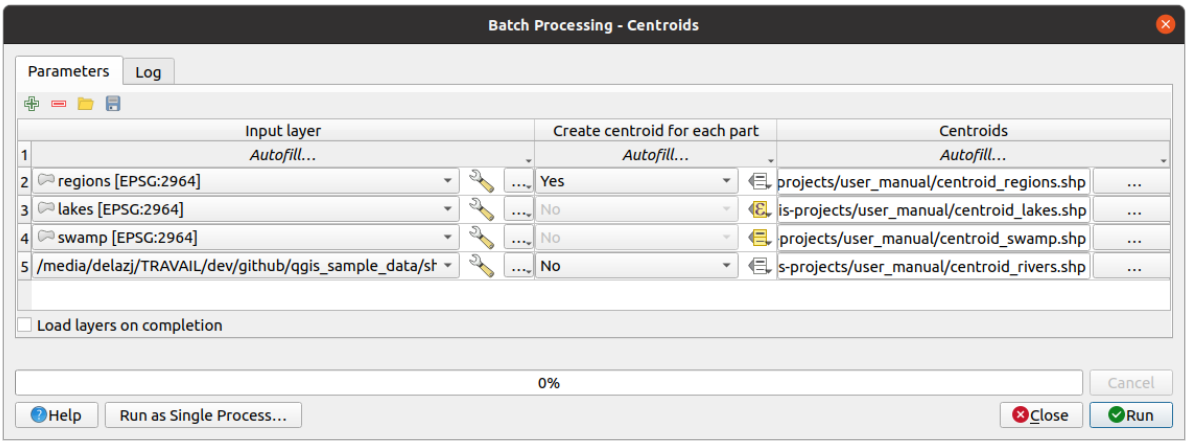


図 26.4: バッチ処理インターフェイス

以下のセクションではそれぞれの要素を詳しく紹介します。

26.2 プロセシングフレームワークを設定する

プロセシングオプションメニュー（**設定** **オプション** **プロセシング** タブ）では、アルゴリズムの動作について設定することができます。設定パラメータは、ダイアログの左側で選択できる個別のブロックに構成されています。

26.2.1 一般情報

一般情報 ブロックには、アルゴリズムダイアログや入力・出力パラメータの動作を制御するためのデフォルト設定が含まれています。しかし、いくつかの設定はアルゴリズム実行時に上書きすることができます。**個々のパラメータ** を参照。

- **デフォルトのラスタレイヤ拡張子** はデフォルトで `tif`
- **デフォルトのベクタレイヤ拡張子** はデフォルトで gpkg
- アルゴリズムを実行するときの **無効地物フィルタ**:
  - **フィルタリングしない（パフォーマンスを向上させる）**：すべての地物（有効と無効なジオメトリを含む）を処理するが、ジオメトリの無効が演算に与える影響により、結果が誤っている場合がある
  - **不正なジオメトリの地物を無視**、つまり、データセットの一部（有効なジオメトリの地物）のみが処理されることを意味する
  - **ジオメトリが無効な場合にアルゴリズムの実効を停止する**: アルゴリズムでレイヤ全体を処理したい場合は無効なジオメトリを追跡して修正する必要があります。**有効性チェック** や **ジオメトリを修復** のようなアルゴリズムが、これを実現するのに役立ちます。

無効地物フィルタ の設定は、アルゴリズムの実行時に入力ごとに上書きすることができます。

- **アルゴリズムの実行時にダイアログを開いたままにする**。アルゴリズムの実行が終了し、その出力レイヤが QGIS プロジェクトに読み込まれると、アルゴリズムダイアログは閉じられます。このダイアログを開いたままにしたい場合（異なるパラメータでアルゴリズムを再実行したり、ログタブに書き込まれる出力をよりよく確認するため）、このオプションをチェックします。
- **最大スレッド数**
- 一時的ではない出力のための **出力フォルダ**: プロセシング実行時の出力にフォルダパスが与えられていない場合、このフォルダに保存されます。デフォルトは、アクティブな user profile ディレクトリ下の `processing/outputs` です。
- **一時出力先パスの上書き**: 一時的な出力は、デフォルトではマシンの `tmp` フォルダーに保存されます。このオプションを使用すると、別の保存場所を設定することができます。
- **事前実行スクリプト** と **事後実行スクリプト**。これらのパラメータは、スクリプトとコンソールのセクションで説明した、プロセシングスクリプト機能を使って書かれたスクリプトを含むファイルを指します。
- **レイヤ名にファイル名を優先使用する**。アルゴリズムによって作成される各結果レイヤの名前は、そのアルゴリズム自体によって定義されます。ある場合には、どの入力レイヤを使用しても同じ出力名が使用されることを意味する、固定名が使用されるかもしれません。他の場合、名前は入力レイヤの名前やアルゴリズムの実行に使われるパラメーターの一部に依存するかもしれません。このチェックボックスをオンにすると、代わりに出力ファイル名から名前が取られます。出力が一時ファイルに保存される場合、この一時ファイルのファイル名は通常、他の既存のファイル名との衝突を避けるための長くて無意味なものであることに注意してください。
- **結果グループ名**。すべてのプロセシング結果レイヤを **レイヤ** パネルでグループ化して取得したい場合、このパラメータにグループ名を設定します。グループは既に存在していてもいなくてもかまいません。QGIS はすべての出力レイヤをこのようなグループに追加します。デフォルトでは、このパラメータは空なので、すべての出力レイヤは、アルゴリズムを実行するときにアクティブになる項目によって、**レイヤ** パネル内の異なる場所に追加されます。なお、出力レイヤが **レイヤ** パネルに読み込まれるのは、アルゴリズムダイアログで **アルゴリズムの終了後に出力ファイルを開く** がチェックされている場合のみです。
- **既知の問題があるアルゴリズムを表示**: デフォルトでは、QGIS は壊れたアルゴリズム（一般的にサードパーティプロバイダのもの）の表示を避けています。チェックした場合、それらはプロセシングツールボックスで利用可能になり、警告アイコンと問題があることを説明するツールチップを表示します。ご自身の責任においてご利用ください。
- **出力レイヤの地物数を表示**。データ形式によっては地物数の計算に時間がかかることがあるため、このオプションはデフォルトでオフになっています。
- **選択ボックス内にレイヤ *CRS* 定義を表示**
- **無効のプロバイダがある場合にツールチップを表示**
- **線レイヤのスタイル**、**点レイヤのスタイル**、**ポリゴンレイヤのスタイル**、**ラスタレイヤのスタイル** は、出力レイヤ（つまり、プロセシングアルゴリズムが生成するレイヤ）に対するデフォルトのレンダリングスタイルを設定するためのものです。QGIS を使って必要なスタイルを作成し、それをファイルに保存し、設定にそのファイルへのパスを入力してアルゴリズムがそれを使用できるようにします。プロセシングによってレイヤが読み込まれ、QGIS キャンバスに追加されるたびに、そのスタイルでレンダリングされます。

レンダリングスタイルは、各アルゴリズムとその出力のそれぞれに対して個別に設定できます。ツールボックスでアルゴリズムの名前を右クリックして **出力のためのレンダリングスタイルを編集** を選択するだけです。次の図のようなダイアログが表示されます。

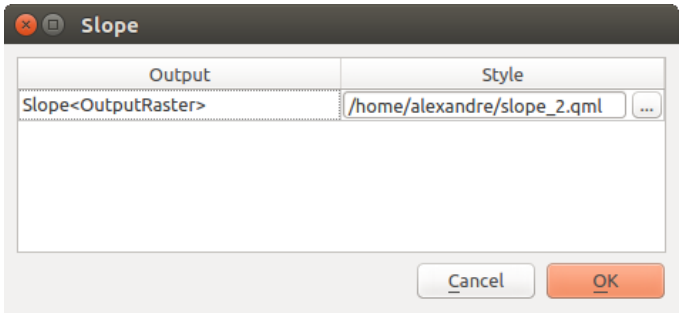


図 26.5: レンダリングスタイル

それぞれの出力に設定したいスタイルファイル (`.qml`) を選択して *OK* を押して下さい。

- **パラメータの *CRS* が合致しない場合は実行前に警告をする**：デフォルトでは、QGIS ネイティブアルゴリズム（ **メニュー**　*QGIS*（**ネイティブ** *C++*） グループにリストされているもの）は、実行前に入力レイヤを最初のレイヤの CRS に透過的に再投影します。このオプションをチェックすると、再投影をサポートしていない他のツールから、入力 CRS が同一でない場合に通知を受け取ることができます。サードパーティプロバイダは関係ありません。

26.2.2 メニュー

メニュー ブロックは、アルゴリズム、スクリプト、モデル（組み込みまたはプラグインによって提供される）を、専用のメニューまたはツールバー（プロセシングツールボックスと共に）で利用できるようにするかどうかを制御します。各プロバイダの各項目について次ができます:

- *Add button in toolbar* は、 **プロセシングアルゴリズム** ツールバーで利用できるようにします
- そのアルゴリズムに *Icon* を割り当てる
- *Menu path* を設定する: このアルゴリズムは、既存のメニューまたはカスタムメニューから利用できるようになります（例: `Vect&or/MyTopAlgorithms` ）

設定を適用するために QGIS を再起動してください。変更した内容は、いつでも **デフォルトにリセットする** ことができます。

26.2.3 モデルとスクリプト

モデル と **スクリプト** ブロックでは、それぞれモデルやスクリプトを保存し、探すためのデフォルトフォルダを設定することができます。

26.2.4 プロバイダ

また、アルゴリズム **プロバイダ** のブロックもあります。ここは、インストールされたプロバイダが設定を公開する場所です。例えば、組み込みプロバイダには **有効化** という項目があり、そのアルゴリズムをツールボックスに表示するかしないかを設定することができます。いくつかのアルゴリズムプロバイダには独自の設定項目があり、特定のアルゴリズムプロバイダを取り上げる際に説明されます。

26.3 ツールボックス

The *Processing Toolbox* is the main element of the processing GUI, and the one that you are more likely to use in your daily work. It shows the list of all available **algorithms** grouped in different blocks called *Providers*, and custom **models** and **scripts** you can add to extend the set of tools. Hence the toolbox is the access point to run them, whether as a single process or as a batch process involving several executions of the same algorithm on different sets of inputs.

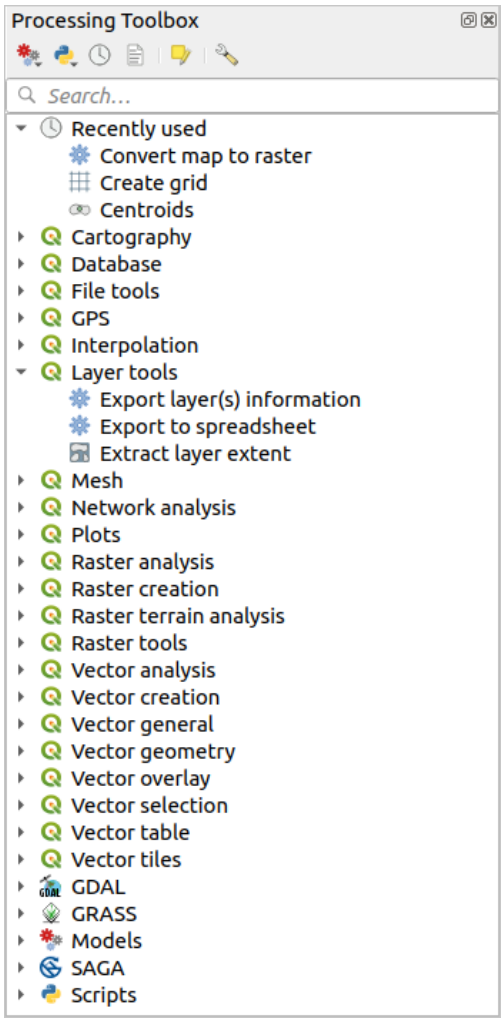


図 26.6: プロセシングツールボックス

プロバイダは **プロセシング設定ダイアログ** で有効化 (無効化) することができます。デフォルトサードパーティアプリケーションに依存しないプロバイダで (QGIS エレメントのみに依存する) が有効です。外部アプリケーションが必要なアルゴリズムについては追加の構成が必要な場合があります。プロバイダの設定についてはこのマニュアルの *later chapter* を参照して下さい.

ツールボックスの上部には次のようなツールがあります:

- work with Models: *Create New Model...*, *Open Existing Model...* and *Add Model to Toolbox...*;
- work with Scripts: *Create New Script...*, *Create New Script from Template...*, *Open Existing Script...* and *Add Script to Toolbox...*;
- open the History panel;
- 結果ビューア パネルを開きます;
- toggle the toolbox to the *in-place modification mode* using the Edit Features In-Place button: only the algorithms that are suitable to be executed on the active layer without outputting a new layer are displayed;
- オプション ダイアログを開きます.

このツールバーの下の **検索...** ボックスはあなたが必要なツールを探す助けになるでしょう. このテキストボックスに任意の単語やフレーズを入力できます. あなたがここでタイプするとツールボックス内のアルゴリズム, モデル, スクリプトのリスト表示があなたが入力したテキストが名前やキーワードに含まれるものだけになるので注意して下さい.

注釈: アルゴリズムリストの上部には最近最も使われたツールの名前が表示されます; 再度実行したい場合に便利です.

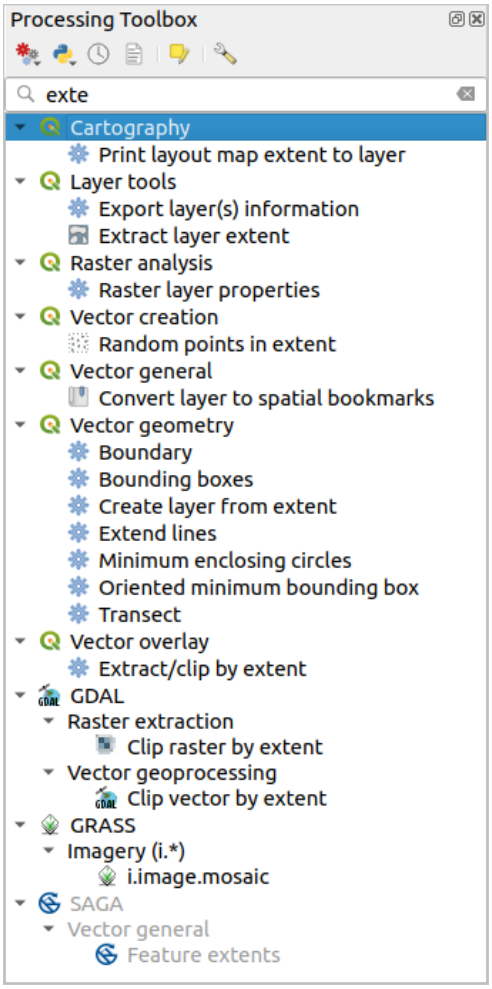


図 26.7: 検索結果を表示している [プロセッシングツールボックス]

ツールを実行するには、ツールボックス内の名前をダブルクリックして下さい.

26.3.1 アルゴリズムダイアログ

実行したいアルゴリズム名をダブルクリックすると、以下の 図 26.8 のようなダイアログが表示されます（この場合は 重心 アルゴリズムに対応したダイアログです）。

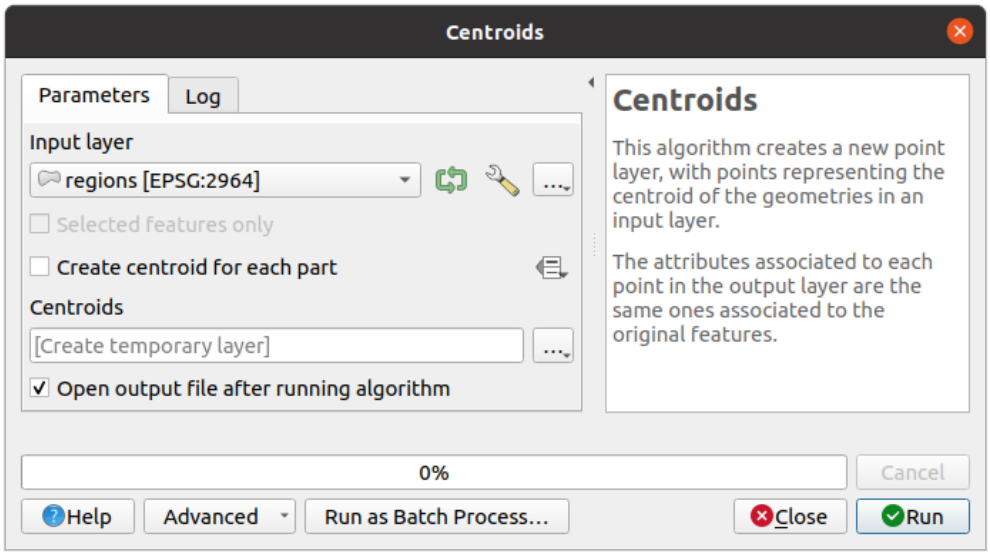


図 26.8: アルゴリズムダイアログ - パラメータ

ダイアログの左側には 2 つのタブ（**パラメータ** と **ログ**）、右側にはアルゴリズムの説明、そして下部には一連のボタンが表示されています。

パラメータのタイプ

パラメータ タブは、アルゴリズムが実行されるために必要な入力値を設定するために使用します。設定すべき入力値や設定パラメータのリストが表示されます。もちろん、実行するアルゴリズムの要件によって異なる内容を持ち、その要件に基づいて自動的に作成されます。

パラメータの数とタイプはアルゴリズムの特性に依存しますが、その構造はすべてのアルゴリズムで似通っています。ここで示されるパラメータは次のいずれかのタイプのみが可能です。

- **ベクタレイヤ** は、QGIS で利用可能な（現在開かれている）すべてのベクタレイヤのリストから選択します。読み込まれていないレイヤを使用することもできます: ウィジェット右側の ... ボタンを押して選択してください：
  - *Select file...*: オペレーティングシステムのファイルエクスプローラーを使ってディスク上のファイルを選択します
  - **レイヤを閲覧**...: **ブラウザパネル** を開き、データベースソース (PostgreSQL, SQL Server, Oracle, ...)、ウェブサービス (WFS, AFS, ...)、ディスク上のファイルから直接レイヤを取得することができます。

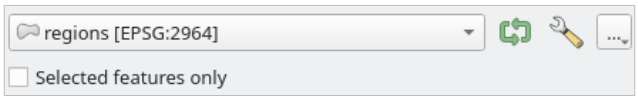


図 26.9: ベクタ入力ウィジェット

注釈: デフォルトでは、レイヤウィジェットはレイヤの CRS をその名前とともに表示します。この追加情報を表示したくない場合は、設定ダイアログの **一般情報** **選択ボックス内にレイヤ** *CRS* **定義を表示** オプションのチェックを外して、この機能を無効にすることができます。

ベクタ入力ウィジェットには次の機能もあります:

- 反復 ボタン： ボタンを切り替えると、アルゴリズムはレイヤ全体に対して 1 回だけ実行されるのではなく、各地物に対して繰り返し実行され、アルゴリズムが実行された回数だけ出力が生成されます。これにより、レイヤ内のすべての地物を個別に処理する必要がある場合に、処理を自動化することができます。アルゴリズムに反復処理できる複数の入力ベクタが含まれている場合、アルゴリズムでパラメータが宣言されている順番で、最初に切り替えられたパラメータに対してのみ反復処理が行われます。
- 詳細オプション ボタンをクリックすると、特定のパラメータに使用する設定を調整できます。これらの設定は以下のことに関係します：
  - **無効地物フィルタ**: 無効なジオメトリの地物を扱う **デフォルトの方法** を上書きします
  - **処理地物を制限**: ソースから処理される地物の数をオプションで制限
- また、ベクタレイヤでのアルゴリズム実行を **選択した地物のみ** に限ることもできます。

- **テーブル**、QGIS で利用可能なすべてのリストから選択します。非空間テーブルはベクタレイヤのように QGIS に読み込まれ、**同じウィジェット** を使います。
- **ラスタレイヤ**、QGIS で利用可能なすべてのラスタレイヤのリストから選択します。セレクタの右側には ... ボタンがあり、QGIS に読み込まれていないレイヤのファイル名を選択することができます。

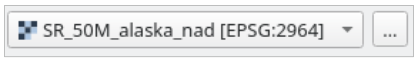


図 26.10: ラスタ入力ウィジェット

- **オプション** , 利用可能なオプションのリストから選択する
- **数値** ではスピンボックスが利用されます. いくつかのコンテキスト (レイヤレベルのものではなく地物レベルに影響するパラメータ) では Data-defined override ボタンが横に表示されます, そこでは *expression builder* を使って数式を入力してパラメータの変数の値を作成することができます. QGIS にロードされたデータに関係するいくつかの便利な変数をあなたの数式に利用することができます, あなたはレイヤのセルのサイズや他のレイヤの最も北の座標のような値を変数として使うことができます.

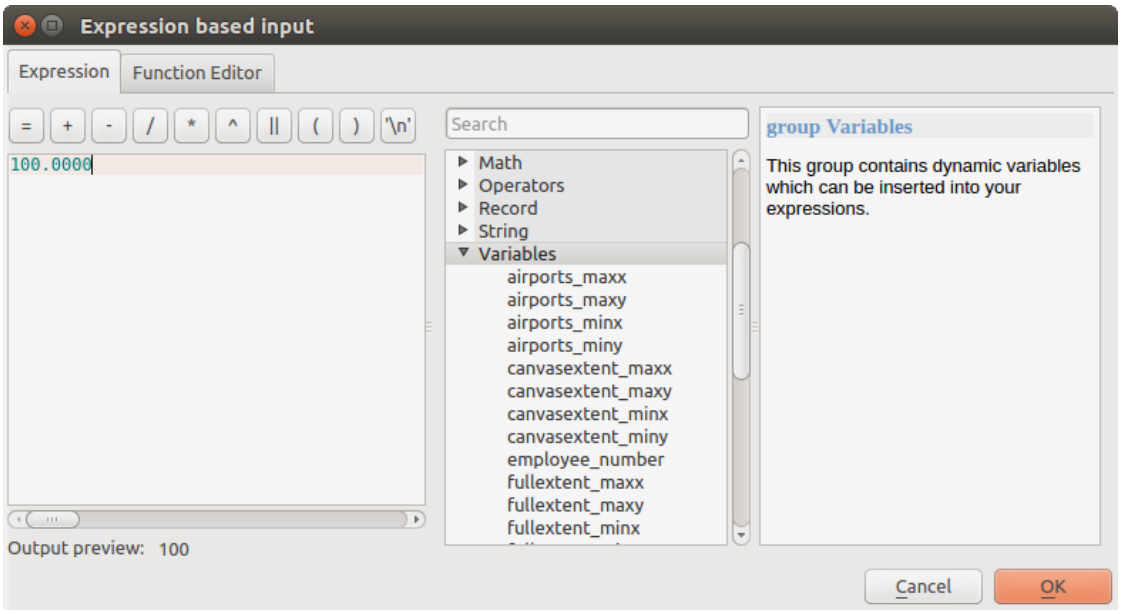


図 26.11: 式に基づく入力

- **範囲**、2 個のテキストボックスで最小値と最大値で指定されます.
- **テキストストリング**、1 個のテキストボックスで指定されます.
- **フィールド**、ベクターレイヤーまたは別のパラメーターで選択された単一のテーブルの属性テーブルから選択します。
- A **coordinate reference system**. You can select it among the recently used ones from the drop-down list or from the *CRS selection* dialog that appears when you click on the button on the right-hand side.
- **範囲**、`xmin, xmax, ymin, ymax` の形式で角の座標から矩形を定義するテキストボックスです。現在のマップキャンバスの範囲を使用するには、  現在のキャンバス領域に設定 ボタンを押してください。値セレクタの右側にある矢印をクリックすると、ポップアップメニューが表示され、次のオプションを選べます:
  - **レイヤから計算**　：読み込まれたレイヤの中から選択したレイヤのバウンディングボックスの座標でテキストボックスを埋めます
  - **レイアウトマップから計算**　：現在のプロジェクトのレイアウトから選択されたマップアイテムの座標でテキストボックスを埋めます
  - **ブックマークから計算**　：保存されたブックマークの座標でテキストボックスを埋めます
  -  **現在のキャンバス領域を利用**
  - **キャンバスに描画**: パラメータウィンドウが非表示になり、キャンバス上でクリック＆ドラッグすることができます。範囲矩形を決めると、ダイアログが再表示され、範囲テキストボックスに値が表示されます。

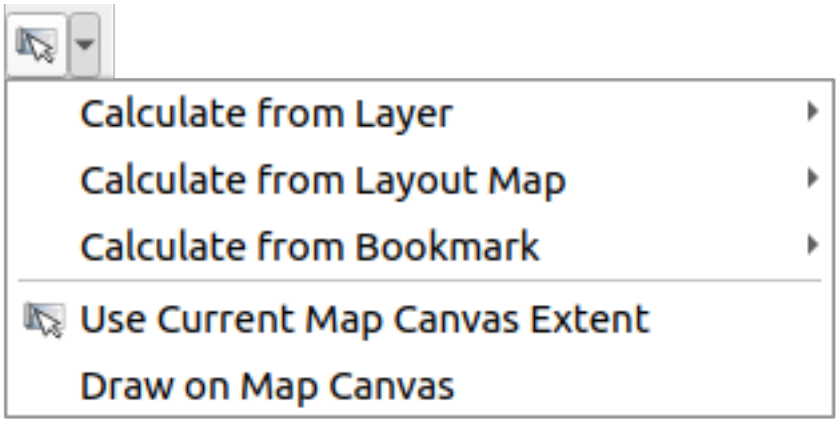


図 26.12: 範囲セレクタ

- A **list of elements** (whether raster or vector layers, tables, fields) to select from. Click on the ... button at the left of the option to see a dialog like the following one. Multiple selection is allowed and when the dialog is closed, number of selected items is displayed in the parameter text box widget.

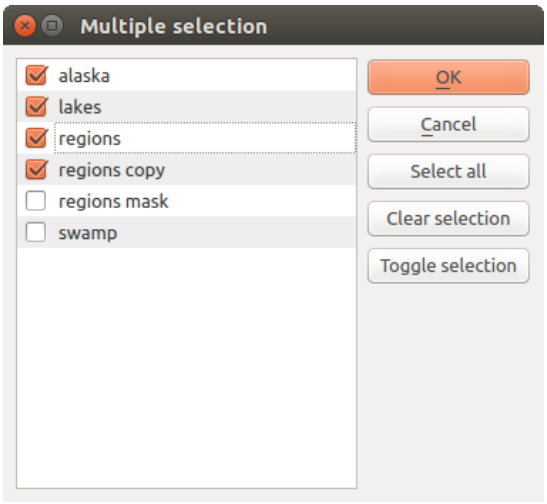


図 26.13: 複数選択

- **小さなテーブル**、ユーザーによって編集されます。これらは、とりわけ、ルックアップテーブルまたはコンボリューションカーネルのようなパラメーターを定義するために使用されます。

  右側にあるボタンをクリックするとテーブルを表示してその値を編集できます.

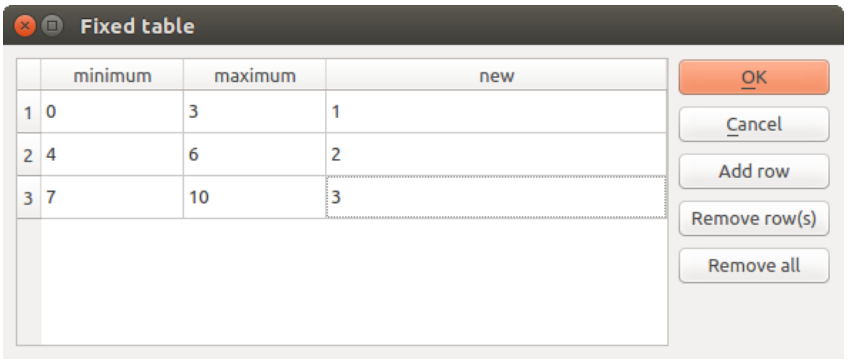


図 26.14: 固定テーブル

アルゴリズムによっては、行数は、ウィンドウの右側にあるボタンを使用せずに変更できます。

注釈: いくつかのアルゴリズムは実行に多くのパラメータを要求します。例えば **ラスタ計算機** では、セルサイズ、範囲、CRS を手動で指定する必要があります。アルゴリズムに 参照レイヤ パラメータがある場合、すべてのパラメータを手動で選択する必要がありません。このパラメータで参照レイヤを選択すると、そのすべてのプロパティ（セルサイズ、範囲、CRS）が使用されます。

実効を記録する

パラメータ タブの他に、**ログ** という名前のタブがあります（下記の 図 26.15 を参照）。このタブには、アルゴリズムが実行中に提供する情報が書き込まれ、実行を追跡できるだけでなく、アルゴリズムが実行されていることを認識し、より詳細な情報を得ることができます。アルゴリズムの実行に関する情報は、**ビュー　パネル　ログメッセージ** にも出力されます。

すべてのアルゴリズムが **ログ** タブに情報を書き込むわけではなく、多くのアルゴリズムは最終的なファイル以外に出力せずに黙々と実行するかもしれないことに注意してください。そのような場合は **ログメッセージ** パネルを確認してください。

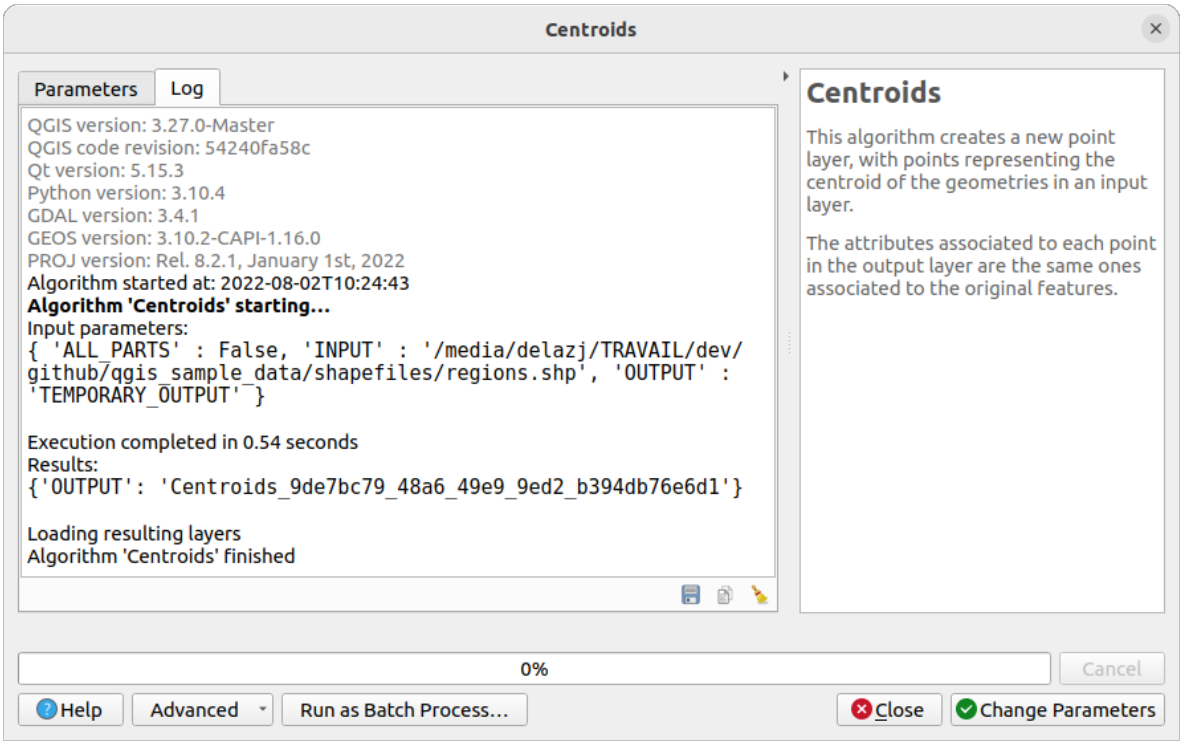


図 26.15: アルゴリズムダイアログ - ログ

ログ タブの下部には **ログをファイルに保存**, **ログをクリップボードにコピー**, **ログをクリア** というボタンがあります。これらは、プロセシングオプションの **一般情報** の **アルゴリズムの実行後にダイアログを開いたままにする** をチェックした場合に特に便利です。

その他のツール

ダイアログの右側にはアルゴリズムの簡単な説明が表示されます。これを読むとこのアルゴリズムの目的と基本的なアイデアが理解できるでしょう。説明が無い場合は説明パネルは表示されません.

ダイアログに下部にある **ヘルプ** ボタンを使うと *Processing algorithms documentation* やプロバイダ (いくつかのサードパーティプロバイダ) の文書で各パラメータの説明や利用例が記述されたさらに詳細なヘルプファイルを読むことができます。

詳細パラメータ　メニューには、アルゴリズムを実行せずにダイアログで定義した設定を再利用するための機能があります:

- *Python* **コマンドとしてコピー**: 等価な *PyQGIS command* を簡単にコピーして、ダイアログに定義したパラメータを使用してツールを実行することができるようにします
- *qgis_process* **コマンドとしてコピー**: 距離単位、面積単位、楕円体などの環境設定や、特定のレイヤを持つ GeoPackage 出力などの厄介なパラメーター値を持った、*qgis_process command* を簡単に生成することができます
- *JSON* **としてコピー**: コマンドのすべての設定が `JSON` 形式でコピーされ、qgis_process で消費できるようになります。これは、複雑なパラメータ（TIN 補間パラメータなど）であっても、コマンドの期

待される形式を確認するのに便利な方法です。これらを簡単に保存しておき、後で値を貼り付けることで復元することができます。

- **設定を貼り付け** は JSON 形式です

バッチプロセスで実効... ボタンを押すと *batch processing mode* が起動し、アルゴリズムの複数のインスタンスを様々なパラメータで構成し実行することができます。**シングルプロセスとして実行**... は、バッチ処理モードから切り替えることができます。

アルゴリズムの実行が終了すると（成功しても失敗しても）、 **ログ** タブがアクティブである限り、新しいボタン **パラメータを変更** が表示されます。

投影法についての注記

プロセシングアルゴリズムは常に入力レイヤの座標参照系 (CRS) で処理を行います。QGIS のオンザフライ再投影機能により、２つのレイヤが重なっているように見えても、同じ座標系に再投影せずに元の座標が使われた場合は、処理の結果が正しくない可能性があります。*QGIS native algorithm* への入力として複数のレイヤを利用する場合はベクタまたはラスタのどちらの場合でも最初の入力レイヤの座標系と同じ座標系に投影されている必要があります。

ただしプロセシングフレームワークを通じて公開される多くの外部アプリケーションではすべてのレイヤが共通の座標系にあり、分析の準備ができていると想定されているためこのことはあまりあてはまりません。

デフォルトでパラメータダイアログに各レイヤの CRS の説明と名前が表示されるので同じ CRS を持つレイヤを選択して簡単に入力レイヤとして利用できます. 追加情報の表示が必要ない場合はプロセッシング設定ダイアログで表示しないように設定できます, *Show layer CRS definition in selection boxes* オプションのチェックをはずして下さい.

もしあなたが CRS が同一でない２つかそれ以上の数の入力を使ってアルゴリズムを実行する場合、警告ダイアログが表示されます。 **パラメータの *CRS* が合致しない場合は実行前に警告する** オプションでこの機能は実行されます。

アルゴリズムは実行できますが、ほとんどの場合、入力レイヤに重なる部分がないために空白レイヤになるなど、間違った結果が生じることに注意してください。

Tip: 再投影を実行してプロセッシングアルゴリズムを利用する

複数の入力レイヤの CRS が一致しなくてアルゴリズムが失敗する場合、**レイヤの再投影** などの QGIS 内部アルゴリズムを使ってレイヤを同じ CRS に再投影し、これらの出力を使ってアルゴリズムを実行します。

26.3.2 アルゴリズムによって生成されるデータオブジェクト

アルゴリズムによって生成されるデータオブジェクトは以下のタイプが利用できます:

- ラスタレイヤ
- ベクタレイヤ
- テーブル
- HTML ファイル (テキストとグラフィック出力の場合利用できます)

これらのデータは全てディスクに保存されます, そしてパラメータテーブルにはレイヤ各出力に対応するテキストボックスが格納されます. 出力チャネルには、結果のオブジェクトをどこかに保存するために必要な情報が含まれています. 多くの場合ファイルに保存するでしょう, しかしベクタレイヤでネイティブアルゴリズムの場合 (外部アプリケーションを使わないアルゴリズム) PostGIS, GeoPackage や SpatiaLite データベース, またはメモリレイヤに保存できます.

出力チャンネルを選択するには、単にテキストボックスの右側にあるボタンをクリックすると、使用可能なオプションを備えた小型のコンテキストメニューが表示されます。

最も一般的なケースでは、ファイルに保存を選択します。そのオプションを選択した場合、目的のファイル・パスを選択でき、保存ファイルダイアログでプロンプトが表示されます。サポートされているファイルの拡張子は出力とアルゴリズムの種類に応じて、ダイアログのファイル形式セレクタに示されています。

出力形式はファイル名の拡張子で定義されます. どのような形式がサポートされているかはアルゴリズムによって異なります. 形式を指定する場合は対応するファイル拡張子を選択すれば可能です (またはファイルパスを直接入力する場合は追加して下さい). ファイルパスで指定された拡張子がサポートしている形式と異なる場合デフォルトの拡張子がファイルパスに利用されます, そしてその拡張子に対応したファイル形式が保存されるレイヤやテーブルに適用されます. デフォルトの拡張子はテーブルについては `.dbf` , ラスタレイヤについては `.tif` ベクタレイヤについては `.gpkg` が利用されます. それらについては設定ダイアログで変更できます, QGIS でサポートされているその他の任意の形式が選択できます.

アウトプットテキストボックスにファイル名を入力しなかった場合（またはコンテキストメニューで対応オプションを選択した場合)、結果は *temporary file* としてデフォルトファイル形式で保存されます。そして QGIS を終了すると削除されます（あなたのプロジェクトを一時レイヤを保持した状態で保存するような場合は注意して下さい。）

出力データを置くデフォルトフォルダを指定することができます。設定ダイアログに移動し（ **設定　オプション　プロセシング** メニューで開けます)、**一般情報** グループの中に **出力フォルダ** というパラメータがあります。この出力フォルダがパス無しでファイル名のみを入力した場合 (i.e., `myfile.shp`) のアルゴリズム実行時の出力フォルダになります。

反復モードのベクタレイヤを使用するアルゴリズムを実行する場合、入力されたファイルパスには、ベース名を使用し、反復のインデックスを表す数値を付加命名されているすべての生成されたファイルのためのベースパスとして使用されます。ファイルの拡張子（と形式）このようなすべての生成されたファイルに使用されます。

アルゴリズムはラスタレイヤとテーブルの他にグラフィックスとテキストを HTML ファイルとして作成します。これらの結果はアルゴリズムの実行後に新しいダイアログで表示されます。このダイアログは現在のセッション実行中は任意のアルゴリズムで作成された結果を保持していて QGIS のメインメニューで **プロセシング　結果ビューア** を選択するといつでも表示できます。

外部アプリケーションには（特に拡張子に制限がない）複数のファイルを出力とするものもありますが、それらは上記のカテゴリのいずれにも属しません。これらの出力ファイルは QGIS によって処理される（開かれたり、現在の QGIS プロジェクトに含まれる）ことはありません、なぜならほとんどの場合それらは QGIS でサポートされていないファイル形式や要素に対応しているからです。これは、例えば、レーザー測量データに使用される LAS ファイルの場合です。ファイルは作成されますが、QGIS 作業セッションには何も新しいものは表示されません。

他の出力タイプのすべてに対し、アルゴリズムによってか生成されると、ファイルをロードするかどうかをアルゴリズムに指示するために使用できるチェックボックスがあります。デフォルトでは、すべてのファイルが開かれます。

オプションの出力はサポートされていません。つまり、すべての出力が作成されます。しかし、与えられた出力に興味がない場合は、対応するチェックボックスのチェックを外すことで、本質的にそれをオプションの出力のように動作させることができます（つまり、レイヤーがとにかく作成されるが、テキストボックスが空のままにした場合、それは一時ファイルに保存され、QGIS を終了した後に削除されます）。

26.4 履歴マネージャ

26.4.1 プロセシングの履歴

アルゴリズムを実行するたびに、そのプロセスに関する情報が履歴マネージャに保存されます。実行された日付と時刻が、使用されたパラメータとともに保存されるため、プロセシングフレームワークを使って開発されたすべての作業を追跡・管理し、再現することが容易です。

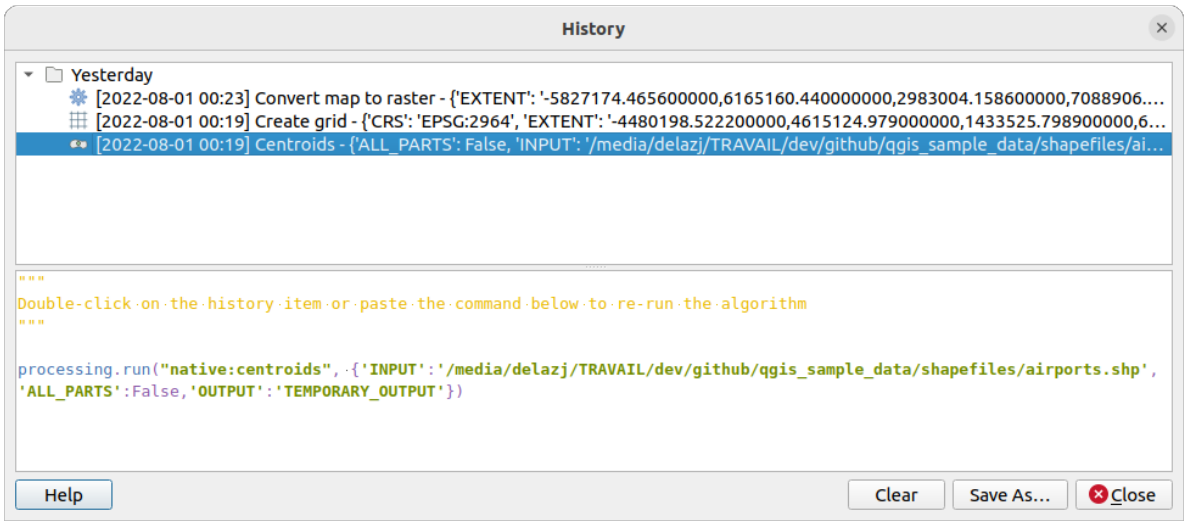


図 26.16: 履歴

プロセス情報は、アルゴリズムがツールボックスから起動されたとしても、コマンドライン式として保持されます。ツールボックスを使ってアルゴリズムを呼び出し、履歴マネージャーをチェックして、コマンドラインからどのように呼び出されたかを確認することができるので、コマンドラインインタフェースの使い方を学んでいる人にとって便利です。

行で右クリックすると次のことができます:

- *Python* **コマンドとしてコピー**: ダイアログから実行した *PyQGIS* **コマンド** と同等のものを簡単にコピーできるようにします。コマンドリストの下に表示されるコードと同じです。
- *qgis_process* **コマンドとしてコピー**: 距離単位、面積単位、楕円体などの環境設定や、特定のレイヤを持つ GeoPackage 出力などの厄介なパラメーター値を持った、:ref:`qgis_process command <processing_standalone>`を簡単に生成することができます
- *JSON* **としてコピー**: コマンドのすべての設定が `JSON` フォーマットでコピーされ、qgis_process で消費できるようになります。これは、複雑なパラメータ（TIN 補間パラメータのような）であっても、コマンドの期待されるフォーマットを確認するのに便利な方法です。これらを簡単に保存し、後でアルゴリズムダイアログに値を貼り付けて復元することができます。
- **テストを作成**... は、:source:`Processing README ファイル <python/plugins/processing/tests/README.md>`の指示に従って、該当するアルゴリズムとパラメータを使って行われます。

レジストリのエントリをブラウズする以外に、エントリをダブルクリックするだけで、プロセスを再実行することもできます。アルゴリズムダイアログが開き、すでにパラメータが設定されているので、必要なパラメータに変更してアルゴリズムを再実行できます。

:guilabel:`履歴`ダイアログは、QGIS プロセシングアルゴリズムとスクリプトのテスト基盤の統合に貢献する便利な方法も提供します。

26.4.2 プロセシングのログ

履歴ダイアログには、実行呼び出しのみが含まれますが、実行時にアルゴリズムによって生成された情報は含まれません。その情報は QGIS ログに書き込まれます (**ビュー　パネル　ログメッセージパネル**)。

サードパーティーのアルゴリズムは通常、コンソールを介してユーザーと通信するコマンドラインインタフェースを使用して実行されます。このコンソールは表示されませんが、通常、これらのアルゴリズムの 1 つを実行するたびに、そのコンソールの完全なダンプがログに書き込まれます。その情報でログが乱雑になるのを避けるために、設定ダイアログでプロバイダごとにそれを無効にすることができます。

アルゴリズムによっては、たとえ与えられた入力データで結果を出すことができても、データの潜在的な問題を検出すると、警告のためにコメントや追加情報をログに出力します。予期せぬ結果が出た場合は、ログのそれらのメッセージを確認してください。

26.5 モデルデザイナー

モデルデザイナー ではシンプルで使いやすいインターフェースを利用して複雑なモデルを作ることができます. GIS で作業をするときにほとんどの解析作業は単独で存在するのではなく一連の操作の一部として存在します. グラフィカルモデラーを使うと一連の操作を単一のプロセスにまとめることができ, 後で異なるデータセットを入力として実行するときに便利です. 多くのステップや異なるアルゴリズムを含んでいてもモデルは単一のアルゴリズムとして実行されるので時間と労力を節約できます.

モデルデザイナーはプロセシングメニュー (**プロセシング　グラフィカルモデラー**) から開けます.

26.5.1 モデルデザイナーのインタフェース

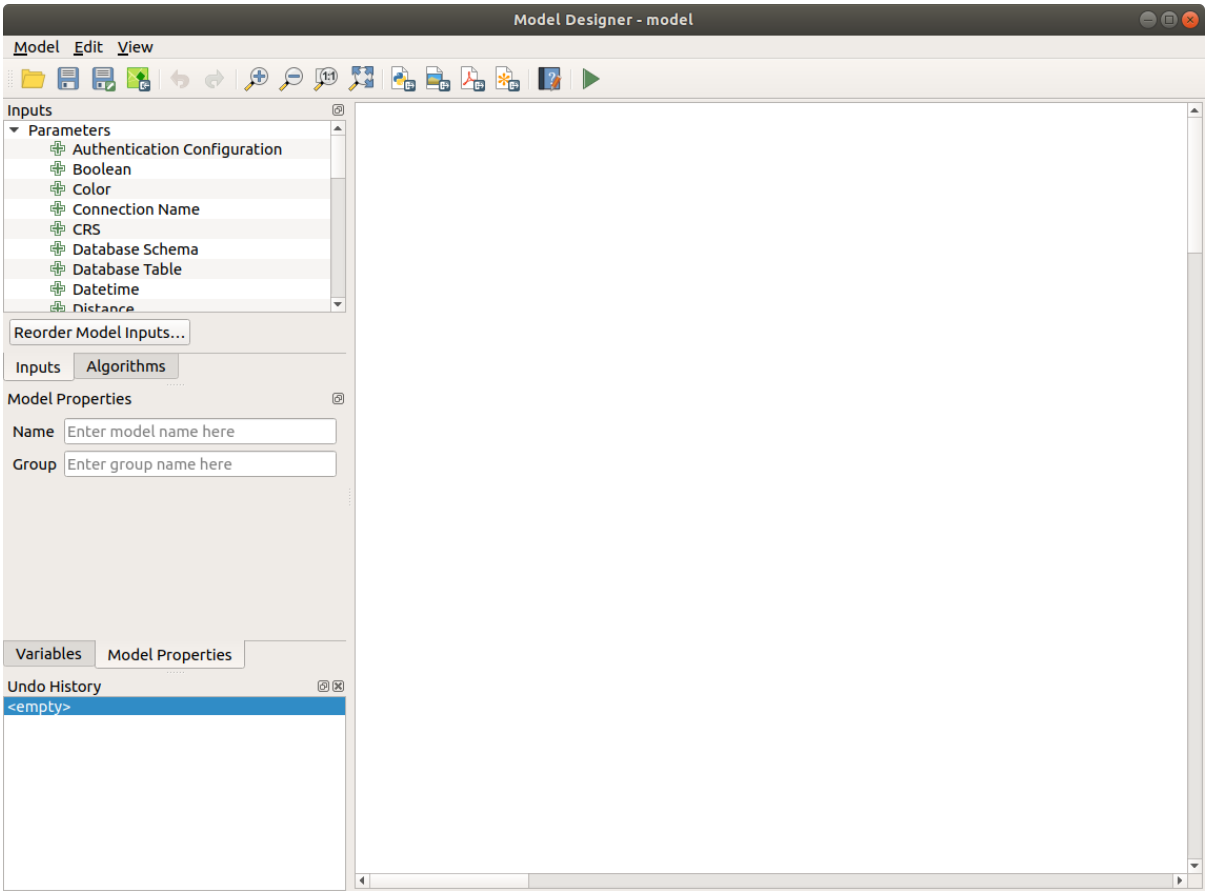


図 26.17: モデルデザイナー

モデラーはその主要部分にモデルの構造やワークフローを構築するための作業用キャンバスが用意されています。

ダイアログの上部には、さまざまなメニューと **ナビゲーション** ツールバーがあり、さまざまなツールにアクセスすることができます。

モデルメニュー

ラベル	ショートカット	ナビゲーションツールバー	説明
モデルを有効化			モデルで使用するアルゴリズムや入力が存在するかどうかをチェックします。モデルをリリースする前に便利です。
モデルを実行...	F5	☑	モデルを実行します
モデル入力の並び替え...			アルゴリズムダイアログでユーザーに提示される入力の順番を設定します。
モデルを開く...	Ctrl+O	☑	編集または実行のために .model3 ファイルを開く
モデルを保存	Ctrl+S	☑	モデルをディスクに .model3 ファイルとして保存する
モデルを別名で保存...	Ctrl+Shif	☑	モデルを新しい .model3 ファイルとしてディスクに保存する
モデルをプロジェクトに保存		☑	モデルファイルをプロジェクトファイルに埋め込み、プロジェクトファイルを共有する際に利用できるようにします。
モデルのヘルプを編集...		☑	モデル、アルゴリズム、パラメータ、出力、さらに作者やバージョン管理などを文書化するインターフェース
エクスポート			
画像としてエクスポート...		☑	モデルのグラフィカルデザインをイメージファイル形式で保存します（図解用）
PDF **として出力...**			モデルのグラフィカルデザインを PDF ファイル形式で保存します（図解用）
SVG **として出力...**			モデルのグラフィカルデザインを SVG ファイル形式で保存します（図解用）
Python **スクリプトとして出力...**		☑	モデルの説明書を含む python スクリプトファイルを生成します

編集メニュー

ラベル	ショートカット	ナビゲーションツールバー	説明
すべて選択	Ctrl+A		デザイナーのすべてのモデルコンポーネントを選択します
選択したコンポーネントをグリッドにスナップ			要素をグリッドにスナップし揃えます
やり直す	Ctrl+Y	☑	最後に取り消されたアクションをロールバックします。**元に戻す/やり直す** パネルも参照すること。
元に戻す	Ctrl+Z	☑	直前の変更を取り消す。**元に戻す/やり直す** パネルも参照すること。
切り取り	Ctrl+X		モデルから選択されたコンポーネントを切り取る。
コピー	Ctrl+C		モデルから選択したコンポーネントをコピーする。
貼り付け	Ctrl+V		切り取りまたはコピーされたコンポーネントの選択部分を、モデルから別のモデル、または同じモデル内に貼り付けます。選択されたコンポーネントは、元のプロパティとコメントを保持します。
選択したコンポーネントを削除	Del		モデルからコンポーネントを削除します。
グループボックスを追加			関連するコンポーネントを視覚的にグループ化するために、関連するコンポーネントの背景にボックスを追加します。特に大きなモデルで、ワークフローをすっきりさせるのに便利です。

ビューメニュー

ラベル	ショートカット	ナビゲーションツールバー	説明
対象ズーム			選択したグループボックスの範囲にズームする
拡大	Ctrl++	☑	
縮小	Ctrl+-	☑	
*100%*にズーム	Ctrl+1	☑	
全域表示	Ctrl+0	☑	デザイナーの現在のキャンバスにあるすべてのコンポーネントを表示します
コメントを表示			グラフィカルデザイナーの各アルゴリズムや入力に関連するコメントを表示する
スナップの有効化			
パネル表示切り替え	Ctrl+Tab		デザイナーの **パネル** を ON や OFF にします

パネル

ウィンドウの左側は、モデルに新しい要素を追加するために使用できる 5 つのパネルがあるセクションです:

1. **モデルのプロパティ**: モデルの名前（必須）と、 **プロセシングツールボックス** に表示するグループを指定します
2. **入力**: モデルを形作るすべての **入力パラメータ**
3. **アルゴリズム**: 利用可能な **プロセシングアルゴリズム**
4. **変数**: モデルには、そのモデルだけが利用できる固有の **変数** を含めることができます。これらの変数は、モデル内で使用されるあらゆる式でアクセスすることができます。この変数は、モデル内でアルゴリズムを制御したり、1 つの変数を変更することでモデルの複数の側面を制御するのに便利です。変数は **変数** パネルで確認・変更することができます。
5. **編集履歴**: このパネルは、モデラーで起こったすべてのことを登録し、間違ったことを簡単に取り消すことができるようにするものです。

利用可能なアルゴリズムについて

ツールボックスから実行できるアルゴリズムの中には、モデルを設計する際に利用可能なアルゴリズムのリストに表示されないものがあります。モデルに含めるには、アルゴリズムが正しいセマンティクスを持っている必要があります。もしアルゴリズムがそのような明確なセマンティクスを持っていない場合（例えば、出力レイヤの数を事前に知ることができない場合)、モデル内で使用することはできず、モデラーダイアログにあるアルゴリズムのリストには表示されません。一方、いくつかのアルゴリズムはモデラーに固有のものです。これらのアルゴリズムは、**モデラーツール** グループにあります。

26.5.2 モデルを作成する

モデルの作成には、基本的に 2 つのステップがあります:

1. **必要な入力の定義**。これらの入力はパラメータウィンドウに追加され、ユーザーはモデルを実行する際にその値を設定することができます。モデル自体はアルゴリズムなので、プロセシングフレームワークで利用可能なすべてのアルゴリズムと同様に、パラメータウィンドウは自動的に生成されます。
2. **ワークフロー** の定義。ワークフローは、モデルの入力データを用い、アルゴリズムを追加し、定義された入力やモデル内の他のアルゴリズムによって生成された出力をどのように使用するかを選択することによって、定義されます。

入力の定義

最初のステップは、モデルの入力を定義することです。以下の要素は、モデラーウィンドウの左側にある **入力** パネルにあります:

表 26.1: モデル構築のためのパラメータタイプの一覧

注記レイヤ	認証の構成	ブール値	色	接続名
座標演算	座標参照系 (CRS)	データベーススキーマ	データベーステーブル	日付時刻
距離	継続時間	DXF レイヤ	列挙	式
領域	属性によるグループ化（集約）	属性の対応関係	ファイル/フォルダ	ジオメト
地図レイヤ	地図テーマ	行列	メッシュデータセットグループ	メッシュ
メッシュレイヤ	複数入力	数値	点	点群レイ
印刷レイアウト	印刷レイアウトアイテム	範囲 (Range)	ラスタバンド	ラスタレ
スケール	文字列 (String)	TIN 作成レイヤ	ベクタ地物	ベクタフ
ベクタレイヤ	ベクタタイル書き出しレイヤ			

注釈: 入力の上にマウスカーソルを置くと、追加情報を示すツールチップが表示されます。

要素をダブルクリックすると、その特性を定義するためのダイアログが表示されます。パラメータによって、ダイアログには少なくとも 1 つの要素（説明文、ユーザーがモデルを実行するときに表示されるもの）が含まれます。例えば、数値を追加する場合、次図のように、パラメータの説明に加えて、デフォルト値

や有効な値の範囲を設定する必要があります。

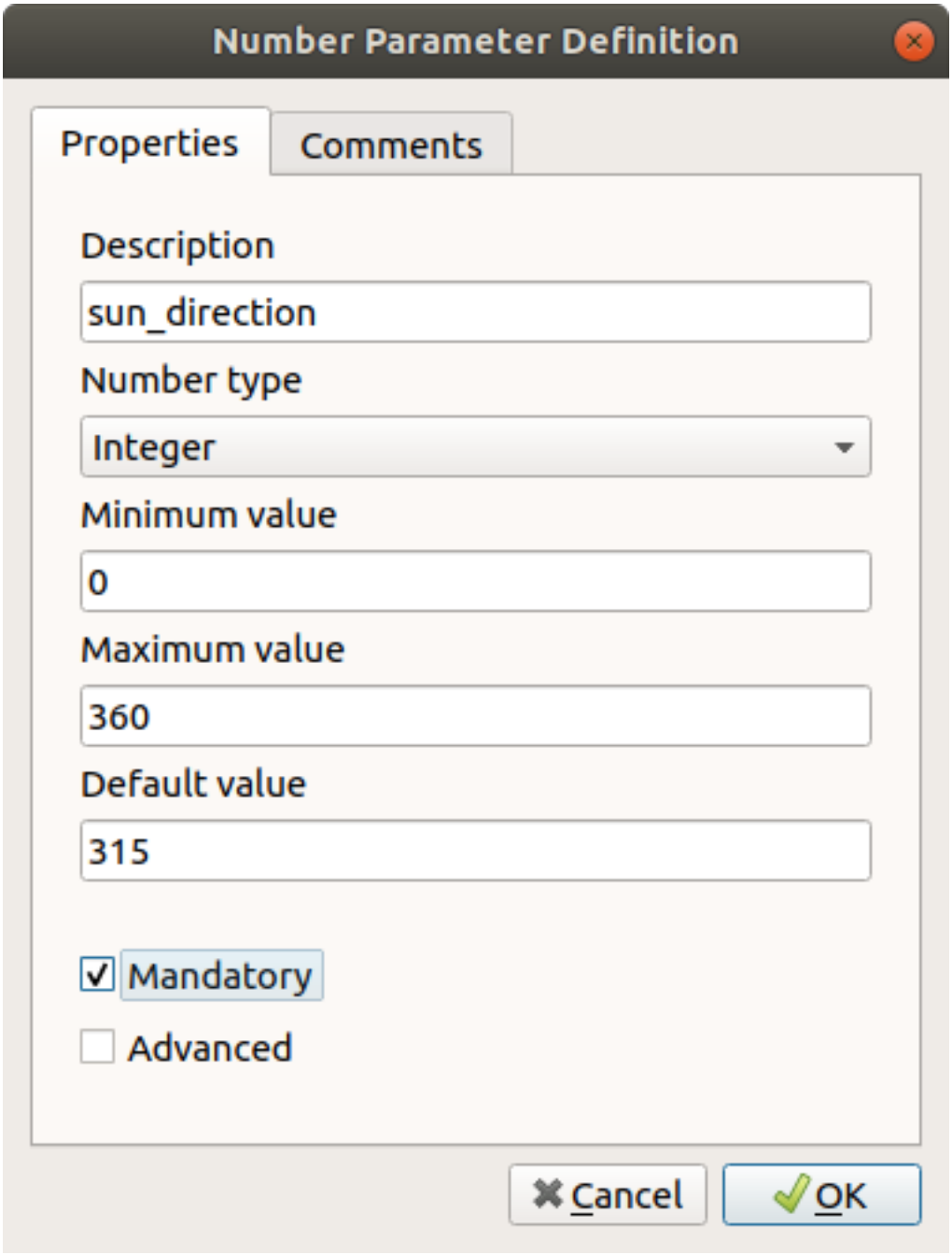


図 26.18: モデルパラメーター定義

チェックボックス| 必須 をチェックすることで、モデルに必須の入力として定義でき、チェックボックス| 詳細設定 をチェックすることで、入力を 詳細設定 セクション内に設定することができます。これは、モデルに多くのパラメータがあり、そのうちのいくつかは些細なものではないが、それでも選択したい場合に特に有効です。

入力が追加されるごとに、モデラーキャンバスに新しい要素が追加されます。

図 26.19: モデルパラメータ

また、リストから入力タイプをドラッグして、モデラーキャンバスの必要な位置にドロップすることで、入力を追加することができます。既存の入力のパラメータを変更したい場合は、その入力をダブルクリックすれば、同じダイアログが表示されます。

他のモデルの中でモデルを使用する場合、必要な入力と出力がキャンバスに表示されます。

ワークフローの定義

次の例では、2 つの入力と 2 つのアルゴリズムを追加します。このモデルの目的は、Drape アルゴリズムを用いて DEM ラスタレイヤからラインレイヤに標高値をコピーし、Climb Along Line アルゴリズムを用いてラインレイヤの合計上昇量を計算することです。

入力 タブで、2 つの入力として、ラインには ベクタレイヤ、DEM には ラスタレイヤ を選択します。これで、ワークフローにアルゴリズムを追加する準備ができました。

アルゴリズムは **アルゴリズム** パネルにあり、プロセシングツールボックスにあるのと同じようにグループ化されています。

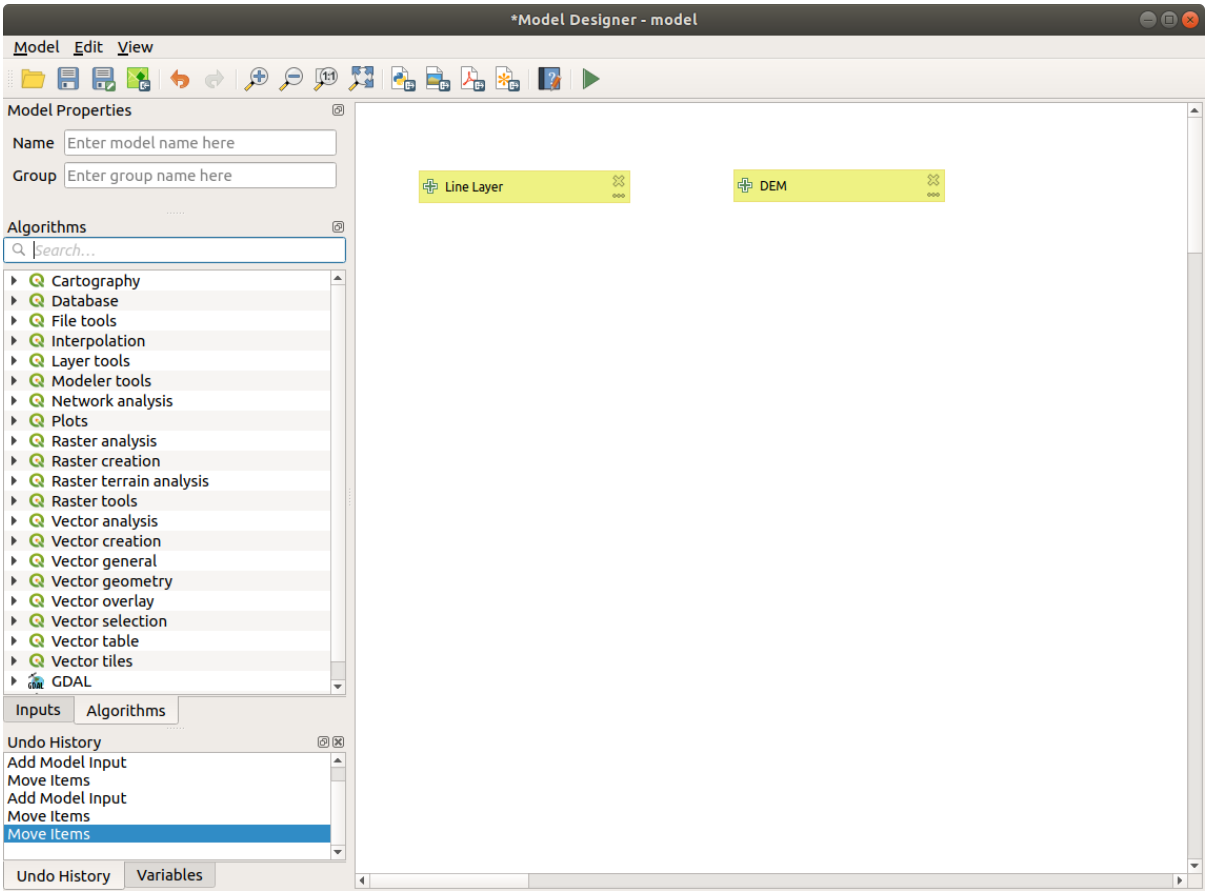


図 26.20: モデル入力

アルゴリズムをモデルに追加するには、入力と同様に、その名前をダブルクリックするか、ドラッグ＆ドロップします。入力の場合と同様に、アルゴリズムの説明を変更したり、コメントを追加したりすることができます。アルゴリズムを追加すると、ツールボックスからアルゴリズムを実行するときに表示される実行パネルと同じような内容の実行ダイアログが表示されます。次の図は、ドレープ（ラスタ値を Z 値に代入） と 線に沿った上昇下降量 の両方のアルゴリズムダイアログを示しています。

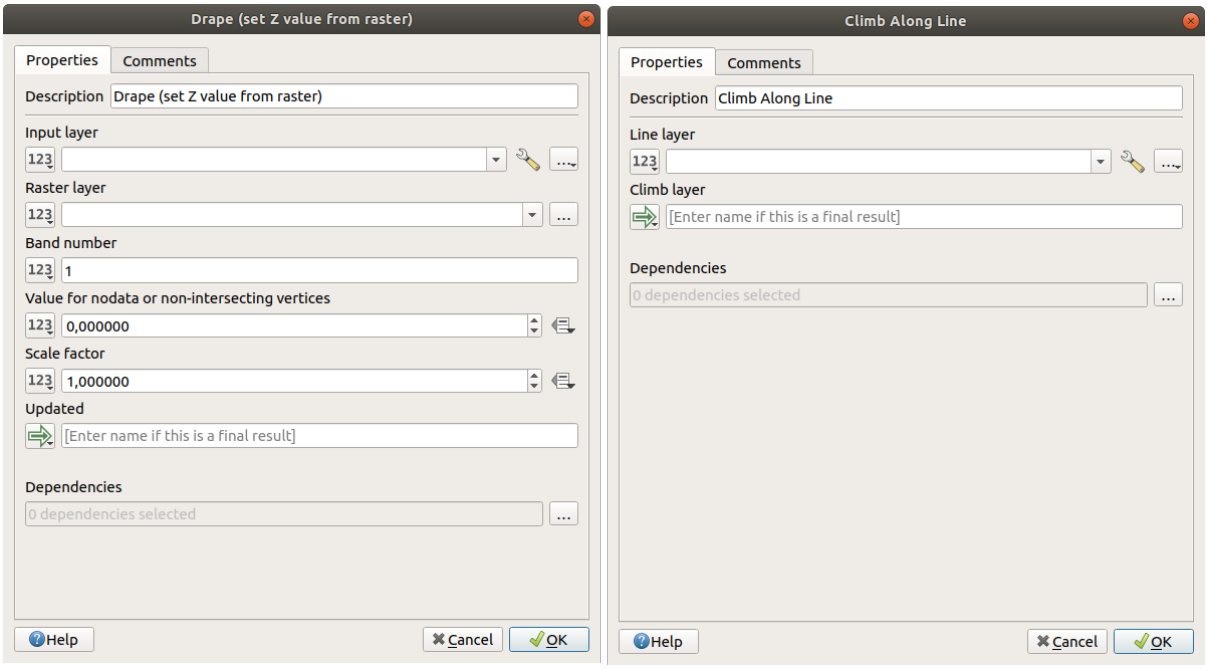


図 26.21: モデルアルゴリズムパラメーター

ご覧のように、いくつかの違いがあります。各パラメータの横にドロップダウンメニューがあり、ワークフロー中にどのように処理されるかをコントロールすることができます:

- 123 値: パラメータに静的な値を割り当てることができます。パラメータタイプに応じて、ウィジェットは数値 (`5.0`)、文字列 (`mytext`) を入力したり、QGIS プロジェクトまたはフォルダからレイヤーを選択したり、リストから項目を選択したりすることができます。
- ε 事前計算済値: **式ビルダ** ダイアログを開き、パラメータを埋める式を定義します。モデル入力と他のレイヤの統計情報は **変数** として利用でき、式ビルダーの検索ダイアログの上部にリストされます。式は子アルゴリズムが実行される前に一度評価され、そのアルゴリズムの実行中に使用されます。
- モデル入力: はモデルに追加された入力をパラメータとして使用することができます。このオプションをクリックすると、パラメータに適した入力がすべてリストアップされます。
- アルゴリズム出力: は、別のアルゴリズムの出力を、現在のアルゴリズムの入力として使用することができます。モデル入力と同様に、このオプションはパラメータに適した入力をすべてリストアップします。
- **出力パラメータ** にも、ドロップダウンメニューに上記のオプションがあります:
  - 子アルゴリズムに静的出力を追加する。例えば、子アルゴリズムの出力を定義済みの geopackage や postgres レイヤに常に保存する
  - 子アルゴリズムに式を基にした出力値を使う。例えば、今日の日付に基づいて自動ファイル名を生成し、そのファイルに出力を保存する
  - モデル入力を使う。例えば、**ファイル/フォルダ** モデル入力を使って、出力ファイルまたはフォルダを指定する
  - 別のアルゴリズム出力を使う。例えば、(**モデラーツール** の) *ディレクトリを作成* アルゴリズムの出力

- 追加の  モデル出力 オプションは、アルゴリズムの出力をモデルで利用できるようにします。アルゴリズムによって生成されたレイヤが他のアルゴリズムの入力としてのみ使用される場合は、このテキストボックスを編集しないでください。

次の図では、モデル入力 として定義された 2 つの入力パラメータと、一時的な出力レイヤを見ることができます:

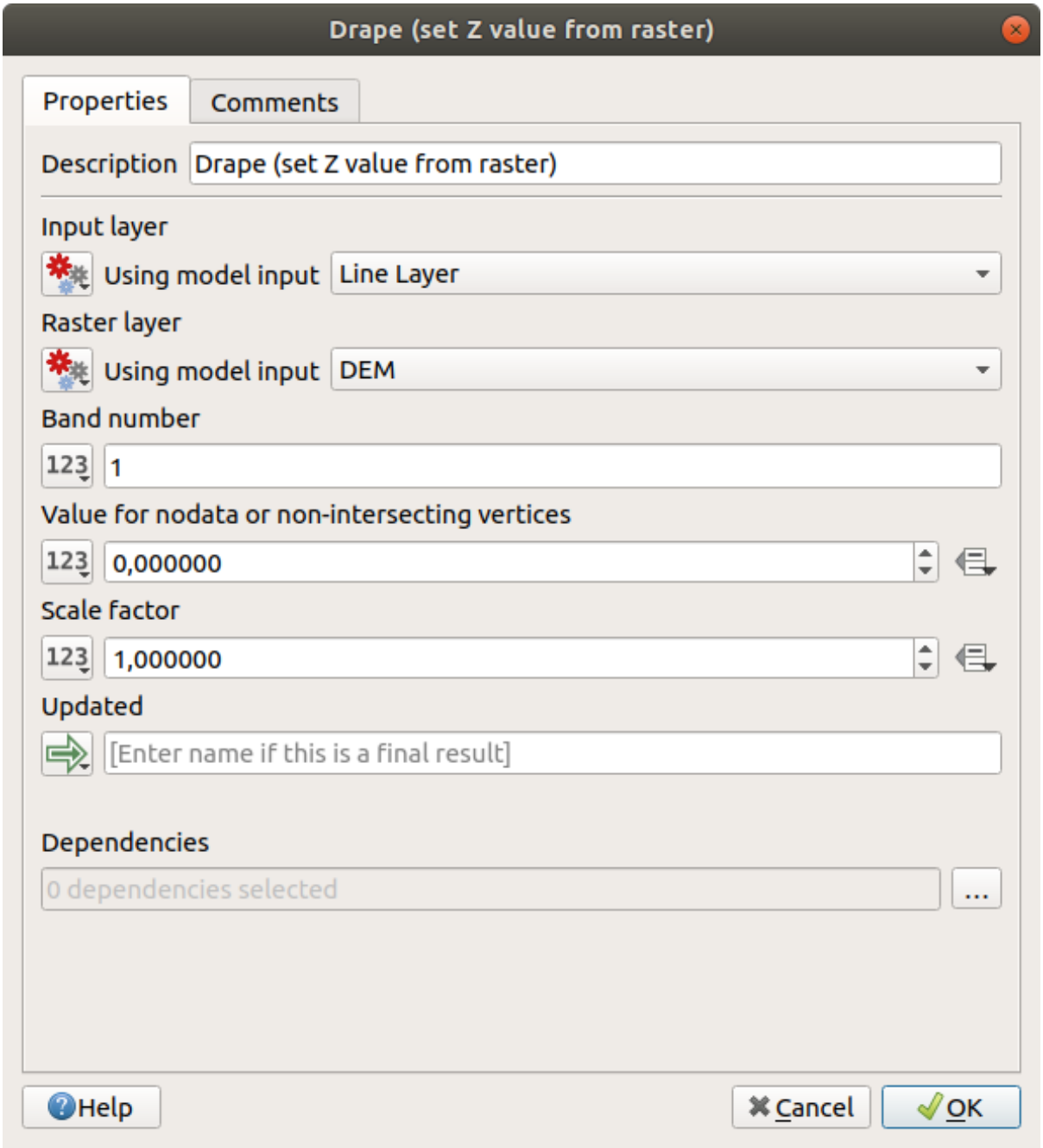


図 26.22: アルゴリズム入力と出力パラメータ

また、ツールボックスからアルゴリズムを呼び出すときには利用できない *Dependencies* という追加のパラメータがあります。このパラメータにより、あるアルゴリズムを現在のアルゴリズムの **親** として明示的に定義することで、アルゴリズムの実行順序を定義することができます。これにより、現在のアルゴリズムの前に **親** アルゴリズムが実行されます。

直前のアルゴリズムの出力を自分のアルゴリズムの入力として使用する場合、暗黙的に直前のアルゴリズムが現在のアルゴリズムの親として設定されます（そしてモデラーキャンバスに対応する矢印が配置されます）。しかし、あるアルゴリズムが、その出力オブジェクトを使用していなくても、別のアルゴリズムに依存している場合があります（例えば、PostGIS データベース上で SQL 文を実行するアルゴリズムと、その同じデータベースにレイヤをインポートする別のアルゴリズム）。その場合、*Dependencies* パラメータで前のアルゴリズムを選択するだけで、正しい順序で実行されます。

すべてのパラメータに有効な値が代入されたら、 *OK* をクリックしてアルゴリズムをキャンバスに追加します。これは、アルゴリズムの入力として使用されるオブジェクトを提供するキャンバスの要素（アルゴリズムまたは入力）にリンクされます。

要素は、 

ツールを使ってキャンバス上の別の位置にドラッグすることができます。これはモデルの構造をより明確に、より直感的にするのに便利です。また、要素の境界をつかんでサイズを変更することもできます。これは入力やアルゴリズムの説明が長い場合に特に便利です。**ビュー　スナップの有効化** オプションをチェックすると、アイテムのサイズ変更や移動が仮想的なグリッドに拘束され、より視覚的に構造化されたアルゴリズム設計が可能になります。

要素間のリンクは自動的に更新され、各アルゴリズムの上部と下部に + ボタンが表示されます。ボタンをクリックすると、そのアルゴリズムのすべての入力と出力が一覧表示されるので、概要を素早く把握することができます。

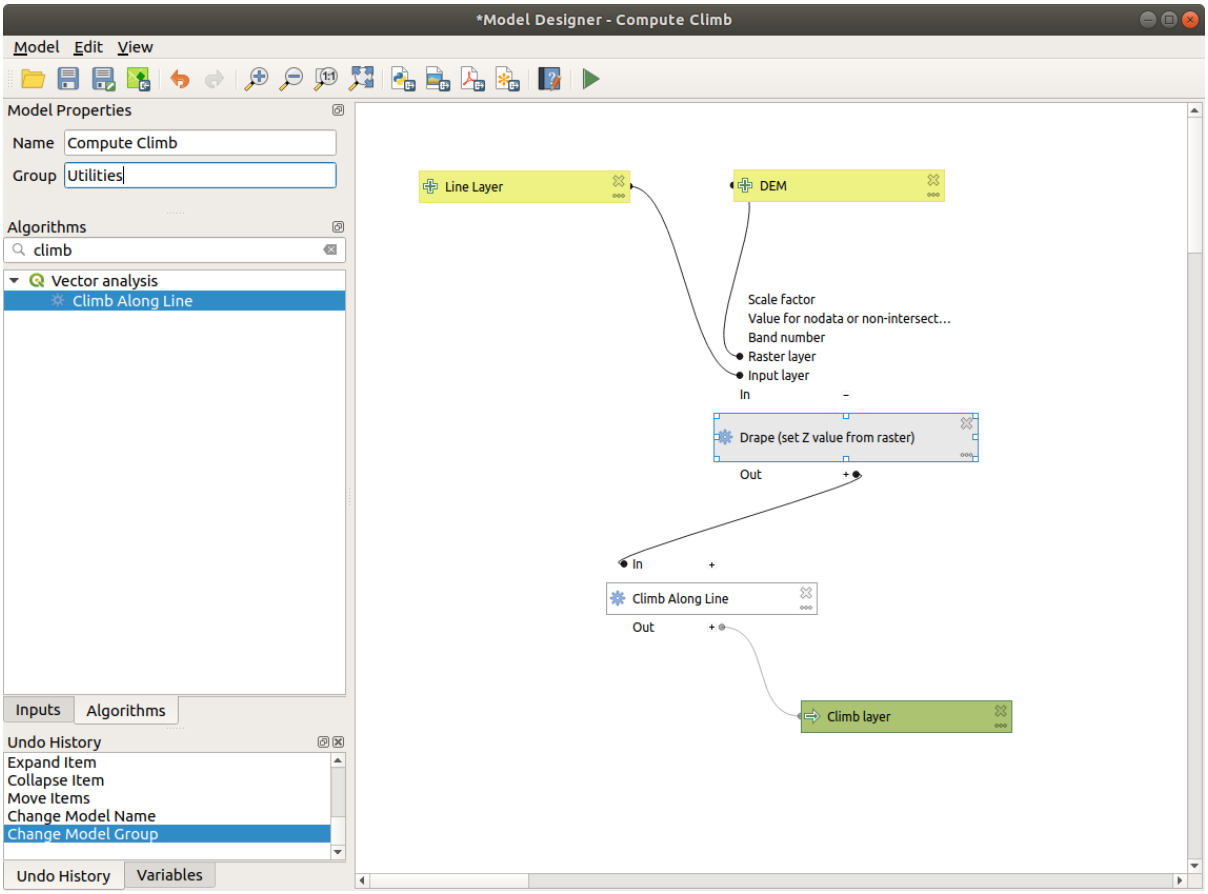


図 26.23: 完成したモデル

編集　グループボックスを追加 ツールを使用すると、キャンバスにドラッグ可能な **ボックス** を追加することができます。この機能は、大きなモデルで関連する要素をモデラーキャンバス内でグループ化し、ワー

クフローをすっきりさせるのに非常に便利です。例えば、例のすべての入力をグループ化することができます:

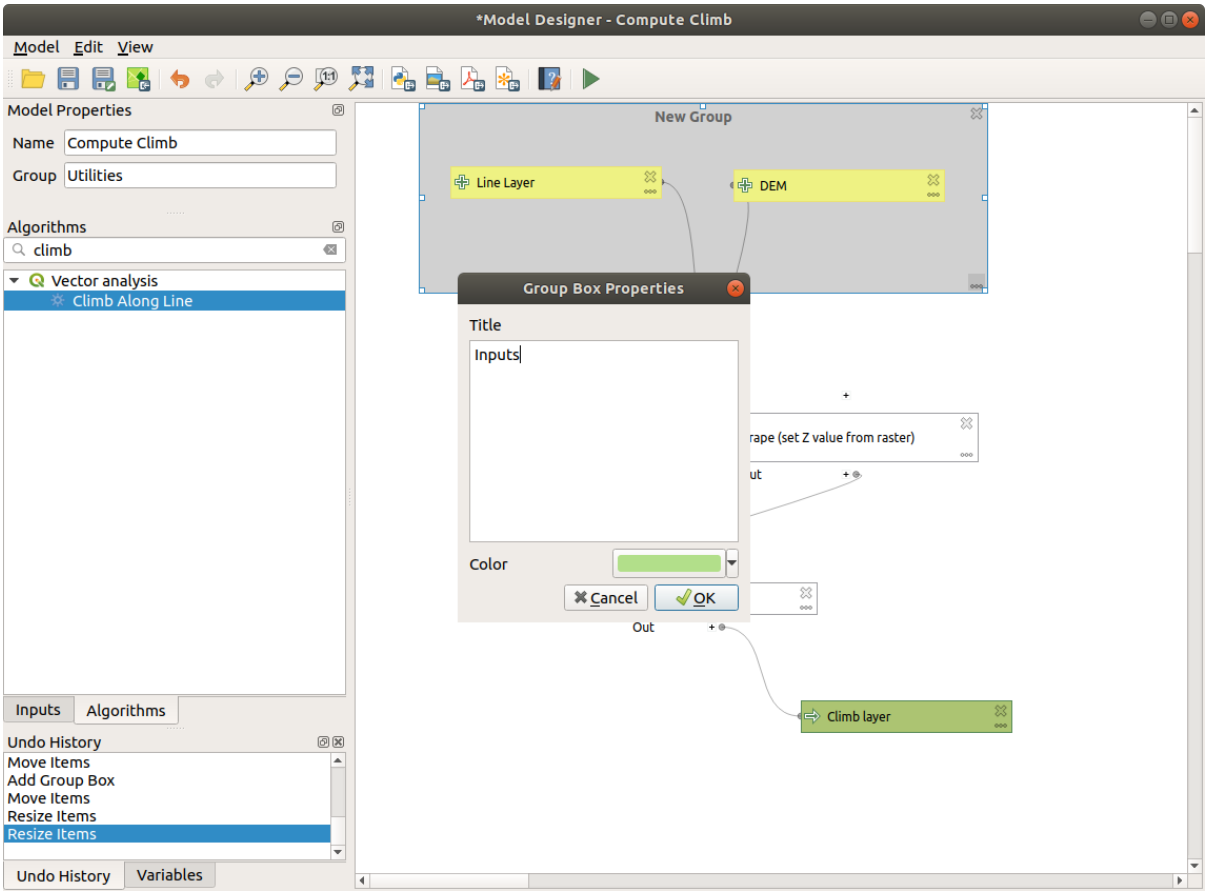


図 26.24: モデルグループボックス

ボックスの名前と色は変更できます。グループボックスは **ビュー** **対象ズーム** ツールと一緒に使うと非常に便利で、モデルの特定の部分にズームインすることができます。また、マウスホイールを使って拡大、縮小することもできます。

入力の順番を変更したり、メインのモデルダイアログにどのように入力が表示されるかを変更したい場 合があるかもしれません。入力 パネルの一番下に``モデル入力の並べ替え... `` というボタンがあり、これをクリックすると新しいダイアログがポップアップ表示され、入力の順番を変更することができます:

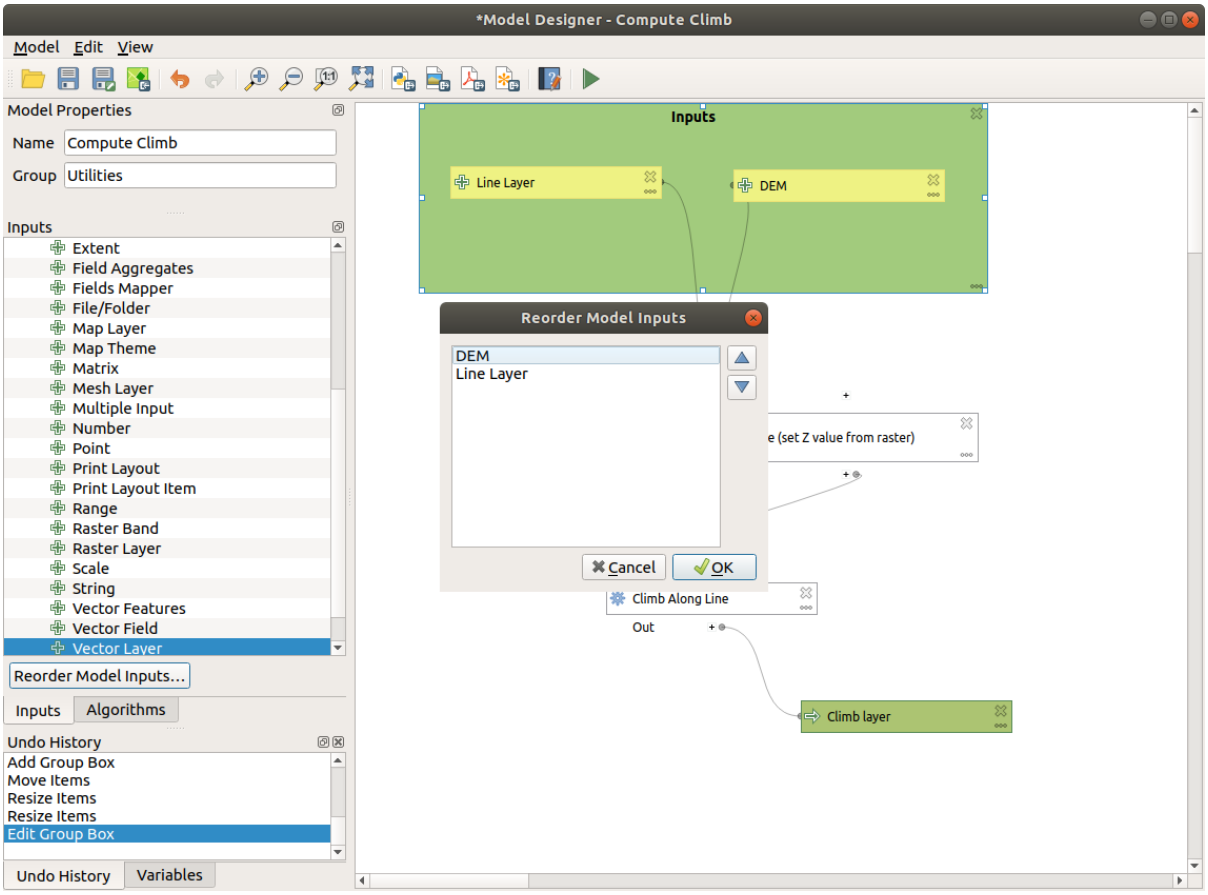


図 26.25: モデル入力の並べ替え

コメントはモデラーにある入力やアルゴリズムに追加することもできます。これはアイテムの **コメント** タブを開くか、右クリックすることで行えます。同じタブで、個々のモデルコメントに対して手動で色を設定することができます。コメントはモデラーキャンバスのみに表示され、最終アルゴリズムダイアログには表示されません。 **ビュー　コメントを表示** を無効にすることで非表示にすることができます。

モデルを実行 ボタンをクリックすることで、いつでもアルゴリズムを実行することができます。モデルを実行するためにエディタを使用するとき、デフォルト以外の値は入力に保存されます。これは、後でエディタからモデルを実行すると、ダイアログにそれらの値が事前に入力されることを意味します。

ツールボックスのアルゴリズムを使用するには、保存してモデラーダイアログを閉じ、ツールボックスの内容を更新する必要があります。

モデルを文書化する

モデルを文書化する必要がありますが、これはモデラー自身から行うことができます。 モデルのヘルプを編集 ボタンをクリックすると、次に示すようなダイアログが表示されます。

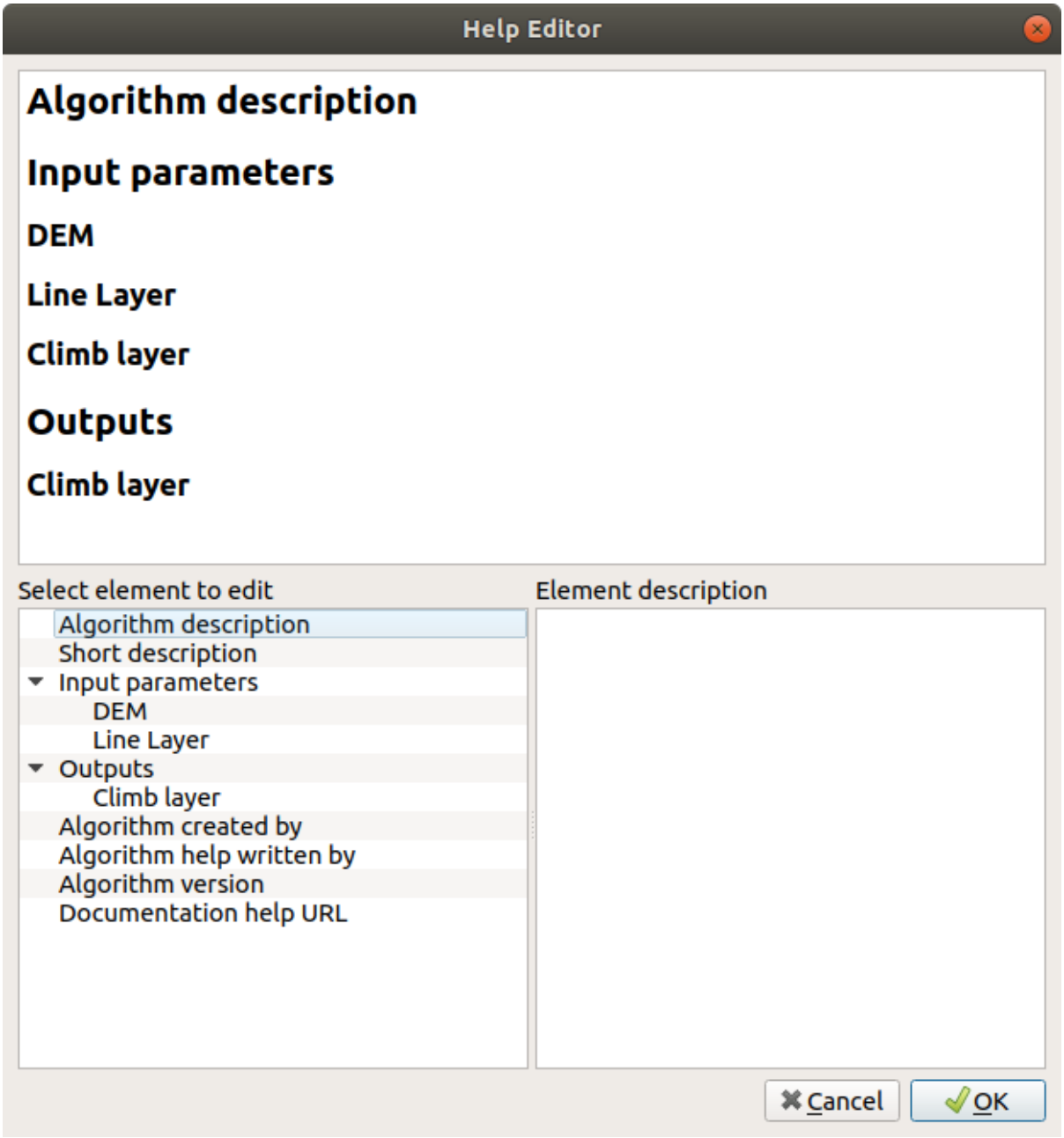


図 26.26: ヘルプを編集する

右側には、単純な HTML ページが表示されます、モデルまたはその作者の一般的な説明のようないくつかの追加のアイテムと一緒に、アルゴリズムの入力パラメーターと出力の記述を使用して作成しました。ヘルプエディタを初めて開いたときには、これらすべての記述が空になっているが、ダイアログの左側にある要素を使用して編集できます。上部の要素を選択し、下のテキストボックスにその説明を記述します。

モデルヘルプはモデル自体の一部として保存されます。

26.5.3 モデルを保存したりロードする

モデルを保存する

現在のモデルを保存するには  モデルを保存 ボタンを、以前に保存したモデルを開くには  モデルを開く ボタンを使用します。モデルは `.model3` という拡張子で保存されます。モデルがすでにモデラーウィンドウから保存されている場合、ファイル名を入力するプロンプトは表示されません。モデルにはすでにファイルが関連付けられているため、以降の保存にはそのファイルが使用されます。

モデルを保存する前に、ウィンドウ上部のテキストボックスに名前とグループを入力する必要があります。

`models` フォルダ（モデルを保存するためのファイル名を入力するプロンプトが表示されたときのデフォルトのフォルダ）に保存されたモデルは、対応するブランチのツールボックスに表示されます。ツールボックスが起動されると、 `models` フォルダから `.model3` という拡張子のファイルを探し、その中に含まれるモデルをロードします。モデルはそれ自体がアルゴリズムなので、他のアルゴリズムと同じようにツールボックスに追加することができます。

また、  モデルをプロジェクトに保存 ボタンを使ってプロジェクトファイル内にモデルを保存することもできます。この方法で保存されたモデルはディスク上に `.model3` ファイルとして書き出されるのではなく、プロジェクトファイルに埋め込まれます。

プロジェクトモデルはツールボックスの  *Project models* メニューと **プロジェクト** → **モデル** メニュー項目で利用できます。

モデルフォルダは **モデラー** グループ下、処理設定ダイアログから設定できます。

`models` フォルダからロードされたモデルはツールボックスだけでなく、モデラーウィンドウの **アルゴリズム** タブにあるアルゴリズムツリーにも表示されます。つまり、他のアルゴリズムと同じように、より大きなモデルの一部としてモデルを組み込むことができるのです。

モデルは **ブラウザ** パネルに表示され、そこから実行することができます。

Python スクリプトとしてモデルを出力する

後の章で説明するように、プロセシングアルゴリズムは QGIS Python コンソールから呼び出すことができ、Python を使って新しいプロセシングアルゴリズムを作成することができます。このような Python スクリプトを作成する簡単な方法は、モデルを作成し、それを Python ファイルとしてエクスポートすることです。

これを行うには、モデラーキャンバスで  スクリプトアルゴリズムとして出力... をクリックするか、プロセシングツールボックスでモデルの名前を右クリックして  Python アルゴリズムとしてモデルを出力... を選択します。

モデルを画像、PDF 又は SVG としてエクスポートする

画像として出力、 PDF として出力、 SVG として出力 をクリックすると、モデルを画像、SVG、PDF（図解用）としてエクスポートすることもできます。

26.5.4 モデルを編集する

現在作成中のモデルを編集し、ワークフローや、モデルを定義するアルゴリズムと入力の関係を再定義することができます。

キャンバス内のアルゴリズムを右クリックすると、次に示すようなコンテキストメニューが表示されます：

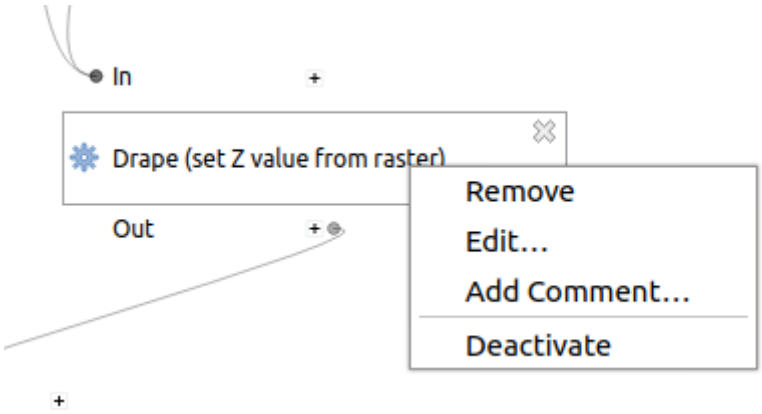


図 26.27: モデラー右ボタンクリック

削除 オプションを選択すると選択されているアルゴリズムを削除できます。アルゴリズムは他のアルゴリズムからの依存がなければ削除できます。つまり、アルゴリズムからの出力が別のアルゴリズムの入力として使用されていない場合。他のアルゴリズムからの依存があるアルゴリズムを削除しようとすると、以下のような警告メッセージが表示されます:

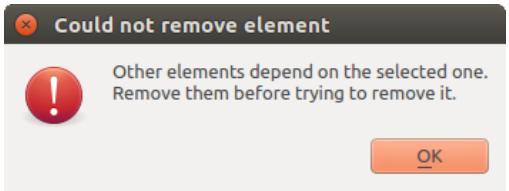


図 26.28: アルゴリズムを削除できない

編集... オプションを選択すると、アルゴリズムのパラメータダイアログが表示され、入力やパラメータの値を変更することができます。モデルで利用可能なすべての入力要素が利用可能な入力として表示されるわけではありません。モデルで定義されたワークフローのより高度なステップで生成されたレイヤや値は、循環依存関係を引き起こす場合には利用できません。

新しい値を選択し、通常通り *OK* ボタンをクリックします。モデル要素間の接続は、それに応じてモデラーキャンバスで変更されます。

コメントを追加... を使うと、アルゴリズムにコメントを追加して動作をより分かりやすく説明することができます。

モデルを部分的に実行するには、いくつかのアルゴリズムを非アクティブにします。これを行うには、アルゴリズム要素を右クリックしたときに表示されるコンテキストメニューの **無効化** オプションを選択します。選択されたアルゴリズムと、それに依存するモデル内のすべてのアルゴリズムが灰色で表示され、モデルの一部として実行されなくなります。

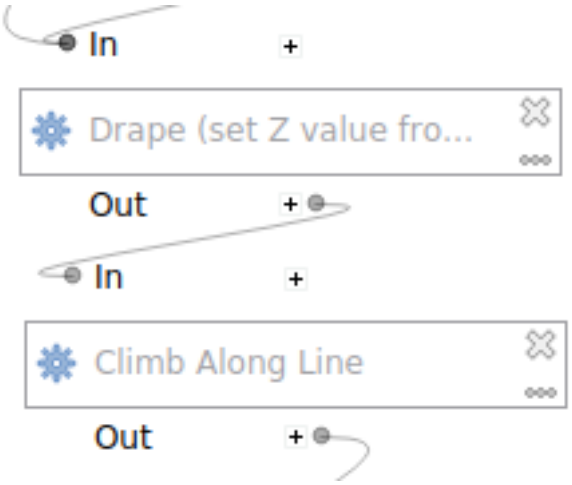


図 26.29: 無効化されたアルゴリズムを持つモデル

アクティブでないアルゴリズムを右クリックすると、 **有効化** メニューオプションが表示されます。

26.6 バッチ処理インターフェイス

26.6.1 はじめに

すべてのアルゴリズム（モデルを含む）はバッチ処理として実行できます。つまりアルゴリズムは、必要な回数を繰り返し実行することで、一つの入力セットだけでなく複数の入力セットを使用して実行できます。大量のデータを処理する際にツールボックスからアルゴリズムを何回も起動する必要がないので、これは便利です。

アルゴリズムをバッチ処理として実行する場合は、ツールボックスの名前を右クリックして表示されるポップアップメニューで、 **バッチプロセスで実行** を選択して下さい。

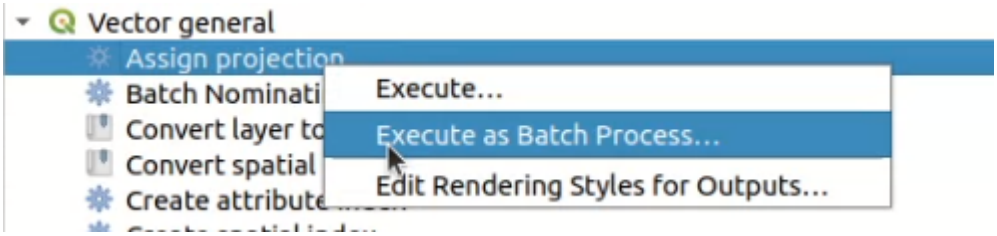


図 26.30: 右クリックからバッチ処理

そのアルゴリズムの実行ダイアログボックスが開いている場合は、**バッチプロセスで実行**... ボタンをクリックすると、そこからもバッチ処理インターフェイスを起動できます。

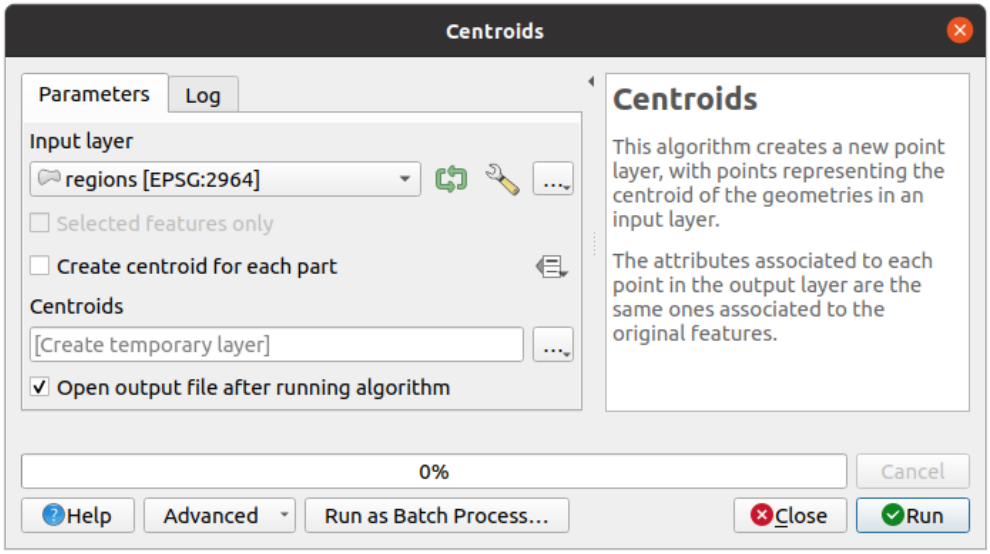


図 26.31: アルゴリズムダイアログからバッチ処理

26.6.2 パラメータテーブル

バッチ処理の実行は、アルゴリズムの一回の実行に似ています。パラメータ値を定義しなければなりませんが、この場合は各パラメータにひとつの値ではなく、アルゴリズムが実行されるたびに一組ずつ使われる、パラメータのセットが必要になります。値は次に示すように、行が反復であり、列がアルゴリズムのパラメータである表を使います。

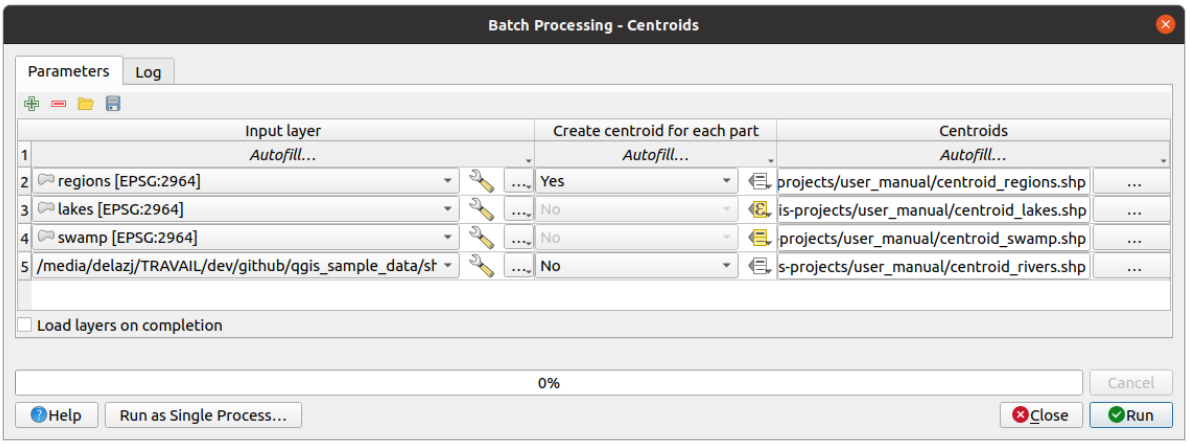


図 26.32: バッチ処理

トップツールバーから：

- 行を追加: 設定に新しい処理項目を追加します

-  行を削除: テーブルから選択した行を削除します。行の選択は左側の数字をクリックすることで行われ、**キーボードとの組み合わせ** で複数選択することができます。
- バッチ処理設定ファイルを  開く
- バッチ処理の設定を .JSON ファイルに  保存 して後で実行できるようにします

表はデフォルトで 2 行だけを含みます:

- 最初の行には、各セルに **オートフィル...** ドロップダウンメニューが表示され、**オプション** を使って下のセルを素早く埋めることができます。利用可能なオプションはパラメータの種類によって異なります。
- 2 行目（およびそれ以降の各行）はアルゴリズムの 1 回の実行を表し、各セルはパラメータの 1 つの値を格納します。これは、ツールボックスからアルゴリズムを実行するときに表示されるパラメータダイアログに似ていますが、配置が異なります。

テーブルの下部では、  **完了時にレイヤを読み込む** かどうかを設定できます。

テーブルのサイズが設定されると適切な値で埋められます.

26.6.3 パラメータテーブルの入力

ほとんどのパラメータでは値の設定は簡単です。**単独のプロセスダイアログ** と同じ適切なウィジェットが提供され、パラメータタイプに応じて、単に値を打ち込むか、可能な値のリストから選択することができます。互換性がある場合には、データ定義ウィジェットも含まれます。

バッチ処理の定義を自動化し、表をセルごとに埋めるのを避けるために、パラメータの **オートフィル...** メニューをドロップして以下のオプションのいずれかを選択し、列の値を置き換えることができます：

- **フィルダウン** は最初のプロセスの入力を受け取り、他の全てのプロセスに入力します。
-  **式による計算...** を使うと、その列内のすべての既存の値を更新するために使用する新しい QGIS 式を作成することができます。既存のパラメータ値 (他の列のものも含む) は *variables* によって式の内部で使用することができます。例. 各レイヤのバッファ距離に基づいてセグメント数を設定する：

```
CASE WHEN @DISTANCE > 20 THEN 12 ELSE 8 END
```

- **式で値を追加...** は、配列を返す式の値を使って新しい行を追加します（既存の行に対してのみ動作する **式による計算...** とは対照的です）。使用例としては、バッチダイアログに複雑な数値列を入力することを想定しています。例えば、距離パラメータに `generate_series(100, 1000, 50)` という式を使ってバッチバッファに行を追加すると、値 100, 150, 200, ... 1000 を持った新しい行になります。
- ファイルやレイヤのパラメータを設定する場合、より多くのオプションが提供されます：
  - **パターンによるファイル追加...**: **対象** フォルダ内の **ファイルパターン** にマッチするファイルのためにテーブルに新しい行を追加します。例えば、`*.shp` はフォルダ内の全ての SHP ファイルをリストに追加します。  **下位フォルダも検索** をチェックすると、サブフォルダも参照することができます。
  - 個別にディスク上の **ファイル選択...**

- **ディレクトリの全ファイルを追加**...
- アクティブなプロジェクトの **レイヤから選択**...

出力データパラメータは、アルゴリズムを単一プロセスとして実行するときと同じ機能を公開します。アルゴリズムによって、出力は以下のようになります：

- セルが空のままならスキップされます
- 一時的なレイヤーとして保存されます: セルを TEMPORARY_OUTPUT で埋め、 **完了時にレイヤを読み込む** チェックボックスにチェックを入れることを忘れないでください。
- プレインファイル（.SHP, .GPKG, .XML, .PDF, .JPG,...）として保存され、そのパスはあらかじめ公開されている **オートフィル** オプションで設定することができます。例えば、**式による計算**... を使用して、出力ファイル名を次のような複雑な式に設定することができます：

```
'/home/me/stuff/buffer_' || left(@INPUT, 30) || '_' || @DISTANCE || '.shp'
```

ファイルパスを直接入力することもできますし、付属の ... ボタンをクリックすると表示されるファイル選択ダイアログを使うこともできます。ファイルを選択すると、新しいダイアログが表示され、同じ列（同じパラメータ）の他のセルを自動入力することができます。

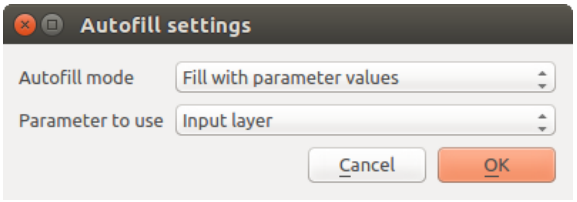


図 26.33: バッチ処理を保存

もしデフォルトの値（**オートフィルしない**）が選ばれた場合は、パラメータテーブルから選択されたセルの中の選択されたファイル名が選ばれます。もし、その他のオプションが選ばれた場合は、**以下**の選択された全てのセルは定義された条件に基づいて自動的に入力されます。

- *Fill with numbers*: ファイル名に数値を加算します
- *Fill with parameter values*: パラメータを選択して同じ行の値をファイル名に付加することができます。これは特に入力データオブジェクトに応じて出力データオブジェクトの名前を付けるのに便利です。

- データベースコンテナ内のレイヤとして保存する：

```
# Indicate a layer within a GeoPackage file
ogr:dbname='C:/Path/To/Geopackage.gpkg' table="New_Table" (geom)

# Use the "Calculate By Expression" to output to different layers in a GeoPackage
'ogr:dbname=\'' || @project_folder || '/Buffers.gpkg\' table="' || @INPUT || '_'␣
↪|| @DISTANCE || '" (geom)'
```

26.6.4 バッチ処理を実行する

必要な値をすべて導入したら、バッチ処理を実行するには **実行** をクリックするだけです。**ログ** パネルがアクティブになり、実行プロセスの詳細とステップが表示されます。グローバルバッチタスクの進行状況は、ダイアログ下部のプログレスバーに表示されます。

26.7 プロセシングアルゴリズムをコンソールから使う

コンソールを使うことによって、上級ユーザは生産性を向上させるとともに、プロセシングフレームワークの他のどの GUI 要素を使っても実現できない、複雑な操作ができるようになります。複数のアルゴリズムを必要とするモデルを、コマンドラインインターフェイスを使って定義することができます。さらにループや条件分岐などの処理を加えることによって、より柔軟で強力なワークフローを作りあげることができます。

QGIS にプロセシング専用コンソールはありませんが、プロセシングのコマンドのすべてが QGIS の内臓 *Python* **コンソール** で実行可能です。つまりそれらのコマンドをコンソールでの作業に組み込み、コンソールで可能なその他すべての機能（QGIS API からのメソッドもそこに含まれます）と、プロセシングアルゴリズムをつなぐことができるということです。

Python コンソールで実行することのできるコードは、たとえそれが特定のプロセシングメソッドを呼び出していないとしても、その他のすべてのアルゴリズムと同様に、後からツールボックスやグラフィカルモデラーやその他のあらゆるコンポーネントから呼び出すことのできる新しいアルゴリズムに変換することができます。実際にツールボックスにあるアルゴリズムのいくつかは、簡単なスクリプトです。

このセクションでは、QGIS Python コンソールからプロセシングアルゴリズムを使用する方法と、Python を使ってアルゴリズムを書く方法を見ていきます。

26.7.1 Python コンソールからアルゴリズムを呼び出す

最初にやるべきことは、次のコードでプロセシングの機能をインポートすることです。

```
>>> from qgis import processing
```

さて、基本的にコンソールからできる（興味深い）ことはただ一つ、アルゴリズムの実行です。これは `run()` メソッドを使って実行します。このメソッドは最初のパラメータとして実行するアルゴリズムの名前を取り、アルゴリズムの要求に応じて追加のパラメータを可変個数取ります。つまり、最初に知る必要があるのは、実行するアルゴリズムの名前です。これはツールボックスに表示される名前ではなく、一意のコマンドライン名です。あなたのアルゴリズムに適した名前を見つけるには、`processingRegistry` を使います。コンソールに以下の行を入力してください：

```
>>> for alg in QgsApplication.processingRegistry().algorithms():
        print(alg.id(), "->", alg.displayName())
```

次のようなものが表示されるはずです（見やすくなるようにダッシュを余計に追加しています）。

```
3d:tessellate ---------------> Tessellate
gdal:aspect -----------------> Aspect
gdal:assignprojection -------> Assign projection
gdal:buffervectors ----------> Buffer vectors
gdal:buildvirtualraster -----> Build Virtual Raster
gdal:cliprasterbyextent -----> Clip raster by extent
gdal:cliprasterbymasklayer --> Clip raster by mask layer
gdal:clipvectorbyextent -----> Clip vector by extent
gdal:clipvectorbypolygon ----> Clip vector by mask layer
gdal:colorrelief ------------> Color relief
gdal:contour ----------------> Contour
gdal:convertformat ----------> Convert format
gdal:dissolve ---------------> Dissolve
...
```

これはすべての利用可能なアルゴリズムの ID と対応する名前のリストです。プロバイダ名とアルゴリズム名でソートされています。

アルゴリズムのコマンドラインでの名前が分かったら、次にすべきことはアルゴリズムを実行するための正しいシンタックスを確認することです。これは `run()` メソッドを呼ぶ際に必要なパラメータを確認するということです。

アルゴリズムを詳細に説明してくれるメソッドがあって、アルゴリズムが必要とするパラメータのリストと、生成される結果を記述してくれます。`algorithmHelp(id_of_the_algorithm)` メソッドを使うとこの情報を得ることができます。アルゴリズムの説明用の名前ではなく、ID を使ってください。

例えば `native:buffer` をパラメータとしてこのメソッドを呼び出すと（`qgis:buffer` は `native:buffer` のエイリアスなので同様に使えます）、以下のような説明が表示されます。

```
>>> processing.algorithmHelp("native:buffer")
Buffer (native:buffer)

This algorithm computes a buffer area for all the features in an
input layer, using a fixed or dynamic distance.

The segments parameter controls the number of line segments to
use to approximate a quarter circle when creating rounded
offsets.

The end cap style parameter controls how line endings are handled
in the buffer.

The join style parameter specifies whether round, miter or
beveled joins should be used when offsetting corners in a line.

The miter limit parameter is only applicable for miter join
styles, and controls the maximum distance from the offset curve
```

(次のページに続く)

(前のページからの続き)

```
to use when creating a mitered join.


----------------
Input parameters
----------------

INPUT: Input layer

   Parameter type: QgsProcessingParameterFeatureSource

   Accepted data types:
            - str: layer ID
            - str: layer name
            - str: layer source
            - QgsProcessingFeatureSourceDefinition
            - QgsProperty
            - QgsVectorLayer

DISTANCE: Distance

   Parameter type: QgsProcessingParameterDistance

   Accepted data types:
            - int
            - float
            - QgsProperty

SEGMENTS: Segments

   Parameter type: QgsProcessingParameterNumber

   Accepted data types:
            - int
            - float
            - QgsProperty

END_CAP_STYLE: End cap style

   Parameter type: QgsProcessingParameterEnum

   Available values:
            - 0: Round
            - 1: Flat
            - 2: Square
```

(次のページに続く)

(前のページからの続き)

```
   Accepted data types:
           - int
           - str: as string representation of int, e.g. '1'
           - QgsProperty

JOIN_STYLE: Join style

   Parameter type: QgsProcessingParameterEnum

   Available values:
           - 0: Round
           - 1: Miter
           - 2: Bevel

   Accepted data types:
           - int
           - str: as string representation of int, e.g. '1'
           - QgsProperty

MITER_LIMIT: Miter limit

   Parameter type: QgsProcessingParameterNumber

   Accepted data types:
           - int
           - float
           - QgsProperty

DISSOLVE: Dissolve result

   Parameter type: QgsProcessingParameterBoolean

   Accepted data types:
           - bool
           - int
           - str
           - QgsProperty

OUTPUT: Buffered

   Parameter type: QgsProcessingParameterFeatureSink

   Accepted data types:
           - str: destination vector file, e.g. 'd:/test.shp'
```

(次のページに続く)

(前のページからの続き)

```
           - str: 'memory:' to store result in temporary memory layer
           - str: using vector provider ID prefix and destination URI,
                  e.g. 'postgres:...' to store result in PostGIS table
           - QgsProcessingOutputLayerDefinition
           - QgsProperty

----------------
Outputs
----------------

OUTPUT:  <QgsProcessingOutputVectorLayer>
   Buffered
```

これでアルゴリズムの実行に必要なことはすべて分かりました。すでに触れたように、アルゴリズムは run() を使って実行することができます。シンタックスは次のとおりです:

```
>>> processing.run(name_of_the_algorithm, parameters)
```

パラメータの箇所には実行したいアルゴリズムに応じたパラメータの辞書がきます。algorithmHelp() メソッドでリストアップされていたものをここで使います。

```
>>> processing.run("native:buffer", {'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
```

パラメータが任意のもので、それを使いたくない時は、そのパラメータを辞書に含めないでください。

パラメータが指定されなかった時はデフォルト値が使われます。

パラメータの種別により値を指定する方法は様々です。次に入力パラメータの種別ごとにその値を指定する方法を簡単に見ていきましょう。

- ラスタレイヤ、ベクタレイヤ、テーブル。これらは単純に、使いたいデータオブジェクトの識別名（QGIS TOC が保持している名前）か、もしくはファイル名を文字列で指定してください。ファイル名を指定した場合、対応するレイヤがまだ読み込まれていない場合はファイルを開いて読み込みますが、マップキャンバスには追加されません。レイヤの QGIS オブジェクトインスタンスをすでに取得している場合は、これをパラメータとして渡すこともできます。
- 列挙。アルゴリズムに列挙パラメータがある場合、そのパラメータの値は整数値で入力する必要があります。利用可能なオプションを知るには、上記のように algorithmHelp() コマンドを使用することができます。例えば、 native:buffer アルゴリズムには JOIN_STYLE という列挙があります：

```
JOIN_STYLE: Join style

   Parameter type: QgsProcessingParameterEnum

   Available values:
           - 0: Round
           - 1: Miter
           - 2: Bevel

   Accepted data types:
           - int
           - str: as string representation of int, e.g. '1'
           - QgsProperty
```

この例では、パラメータは 3 つの選択肢を持ちます。値は 0 始まりであることに注意してください。

- 真偽値。True か False を使用してください。
- 複数の入力。値は入力を記述したものをセミコロン (;) で区切った文字列です。各入力が単一レイヤやテーブルの場合はデータオブジェクト名やファイルパスが使えます。
- XXX のテーブル項目名。利用する項目名の文字列を使ってください。このパラメータは大文字小文字を区別します。
- 固定テーブル。テーブルのすべての値のリストを、各値をカンマ (,) で区切った上で、全体を引用符 (") で囲んで入力します。値は最上位列から始まり、左から右に進みます。テーブルを表す 2 次元の配列も使えます。
- CRS。使いたい CRS の EPSG コード番号を入力します。
- 範囲。xmin, xmax, ymin および ymax の値を、カンマ (,) で区切って、文字列で指定しなければなりません。

真偽値、ファイル、文字列および数値パラメータには特に付け加える説明はありません。

文字列、真偽値、数値などの入力パラメータはデフォルト値を持ちます。デフォルト値は対応するパラメータの入力が欠けていた時に使われます。

出力データオブジェクトには、ツールボックスから実行される時と同様に、それを保存するために使うファイルパスを入力してください。出力オブジェクトが指定されない時は、結果は一時ファイルに保存されます (もしくは出力が任意の時はスキップされます)。ファイルの拡張子はファイルフォーマットから決定されます。アルゴリズムでサポートされていないファイル拡張子を入力した場合は、その出力タイプのデフォルトファイルフォーマットが使われ、それに対応する拡張子が与えられたファイルパスに付加されます。

ツールボックスからアルゴリズムを実行した場合とは異なり、run() を使って Python コンソールから同じアルゴリズムを実行した場合、マップキャンバスに出力は追加されませんが、runAndLoadResults() はそれをします。

run() メソッドは、1 つ以上の出力名（アルゴリズムの説明に示されているもの）をキーとし、それらの出力のファイルパスを値とする辞書を返します：

```
>>> myresult = processing.run("native:buffer", {'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
>>> myresult['OUTPUT']
/data/buffers.shp
```

load() メソッドに対応するファイルパスを渡すことで、地物の出力を読み込むことができます。あるいは、 run() の代わりに runAndLoadResults() を使用すると、すぐに読み込むことができます。

コンソールからアルゴリズムダイアログを開きたい場合は、 createAlgorithmDialog メソッドを使います。必須パラメータはアルゴリズム名だけですが、パラメータの辞書を定義することで、ダイアログを自動的に埋めることもできます：

```
>>> my_dialog = processing.createAlgorithmDialog("native:buffer", {
                'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
>>> my_dialog.show()
```

execAlgorithmDialog メソッドはダイアログをすぐに開きます:

```
>>> processing.execAlgorithmDialog("native:buffer", {
                'INPUT': '/data/lines.shp',
                'DISTANCE': 100.0,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10,
                'OUTPUT': '/data/buffers.shp'})
```

26.7.2 スクリプトを作成してツールボックスから実行する

Python コードを書くことによって自分独自のアルゴリズムを書くことができます。プロセシングスクリプトは QgsProcessingAlgorithm クラスを拡張します。そのため必要な機能を実装するためにはそこにさらにコードを追加する必要があります。プロセシングツールボックス最上部の **スクリプト** ドロップダウンメニューから **新しいスクリプトを作成...**（白紙）と **テンプレートから新しいスクリプトを作成...**（必須の関数 QgsProcessingAlgorithm のコードを含むテンプレート）を選ぶことができます。するとコードを入力するためのプロセシングスクリプトエディタが開きます。ここでスクリプトを保存すると scripts フォルダ（スクリプトを保存ダイアログを開いたときのデフォルトフォルダ）に .py 拡張子でアルゴリズムとして保存されます。

アルゴリズムの名前（ツールボックスに表示されるもの）はコードの中で定義されます。

プロセシングアルゴリズムを定義する次のコード例を見てみましょう。このアルゴリズムは、最初のレイヤ平滑化の後に、ユーザによって指定されたベクタレイヤに対する、ユーザ定義のバッファ距離によるバッファ操作を行うアルゴリズムとして機能するものです。

```
from qgis.core import (QgsProcessingAlgorithm,
       QgsProcessingParameterNumber,
       QgsProcessingParameterFeatureSource,
       QgsProcessingParameterFeatureSink)

from qgis import processing

class algTest(QgsProcessingAlgorithm):
    INPUT_BUFFERDIST = 'BUFFERDIST'
    OUTPUT_BUFFER = 'OUTPUT_BUFFER'
    INPUT_VECTOR = 'INPUT_VECTOR'

    def __init__(self):
        super().__init__()

    def name(self):
        return "algTest"

    def displayName(self):
        return "algTest script"

    def createInstance(self):
        return type(self)()

    def initAlgorithm(self, config=None):
        self.addParameter(QgsProcessingParameterFeatureSource(
            self.INPUT_VECTOR, "Input vector"))
        self.addParameter(QgsProcessingParameterNumber(
            self.INPUT_BUFFERDIST, "Buffer distance",
            QgsProcessingParameterNumber.Double,
```

(次のページに続く)

(前のページからの続き)

```
            100.0))
        self.addParameter(QgsProcessingParameterFeatureSink(
            self.OUTPUT_BUFFER, "Output buffer"))

    def processAlgorithm(self, parameters, context, feedback):
        #DO SOMETHING
        algresult = processing.run("native:smoothgeometry",
            {'INPUT': parameters[self.INPUT_VECTOR],
             'ITERATIONS':2,
             'OFFSET':0.25,
             'MAX_ANGLE':180,
             'OUTPUT': 'memory:'},
            context=context, feedback=feedback, is_child_algorithm=True)
        smoothed = algresult['OUTPUT']
        algresult = processing.run('native:buffer',
            {'INPUT': smoothed,
            'DISTANCE': parameters[self.INPUT_BUFFERDIST],
            'SEGMENTS': 5,
            'END_CAP_STYLE': 0,
            'JOIN_STYLE': 0,
            'MITER_LIMIT': 10,
            'DISSOLVE': True,
            'OUTPUT': parameters[self.OUTPUT_BUFFER]},
            context=context, feedback=feedback, is_child_algorithm=True)
        buffered = algresult['OUTPUT']
        return {self.OUTPUT_BUFFER: buffered}
```

必要なインポートを行った後に、続いて QgsProcessingAlgorithm の諸関数が定義されます。

- name(): アルゴリズムの id（小文字）。
- displayName(): アルゴリズムの人が読む名前。
- createInstance(): アルゴリズムクラスの新しいインスタンスを作る。
- initAlgorithm(): parameterDefinitions と outputDefinitions を設定する。

  ここでアルゴリズムのパラメータと出力を記述します。この例では、入力のためには feature source を、出力結果のためには feature sink とバッファ距離用の数値を定義しています。

- processAlgorithm(): 処理を行う。

  ここでは、まず smoothgeometry アルゴリズムを実行してジオメトリを平滑化し、次に平滑化された出力に対して buffer アルゴリズムを実行します。他のアルゴリズムからアルゴリズムを実行するには、 is_child_algorithm 引数を True に設定する必要があります。入力パラメータと出力パラメータが smoothgeometry と buffer アルゴリズムのパラメータとしてどのように使用されているかを見ることができます。

入力と出力のために多くの種類のパラメータが使用可能です。以下はアルファベット順にソートされたリ

ストです。

表 26.2: 入出力アルゴリズムパラメータ型の

QgsProcessingParameterAggregate	QgsProcessingParameterAnnotationLayer	QgsProcessingP
QgsProcessingParameterBoolean	QgsProcessingParameterColor	QgsProcessingP
QgsProcessingParameterDatabaseSchema	QgsProcessingParameterDatabaseTable	QgsProcessingP
QgsProcessingParameterEnum	QgsProcessingParameterExpression	QgsProcessingP
QgsProcessingParameterFeatureSource	QgsProcessingParameterField	QgsProcessingP
QgsProcessingParameterFileDestination	QgsProcessingParameterFolderDestination	QgsProcessingP
QgsProcessingParameterLayoutItem	QgsProcessingParameterMapLayer	QgsProcessingP
QgsProcessingParameterMeshLayer	QgsProcessingParameterMultipleLayers	QgsProcessingP
QgsProcessingParameterPointCloudLayer	QgsProcessingParameterProviderConnection	QgsProcessingP
QgsProcessingParameterRasterLayer	QgsProcessingParameterScale	QgsProcessingP
QgsProcessingParameterVectorLayer	QgsProcessingParameterVectorTileWriterLayers	

コンストラクタの最初のパラメータはパラメータ名です。2 番目はユーザインターフェイス用のパラメータの説明です。それ以外のパラメータは、パラメータの種別に依存します。

入力は QgsProcessingAlgorithm クラスの parameterAs 関数によって QGIS クラスに変換されます。例えばバッファ距離のために数値を double で取得するには次のようにします。

```
self.parameterAsDouble(parameters, self.INPUT_BUFFERDIST, context)).
```

processAlgorithm 関数は、アルゴリズムで定義されたすべての出力値を含む辞書を返さなければなりません。これによって、同一モデル内の別のアルゴリズムを含む他のアルゴリズムから、これらの出力にアクセスすることが可能になります。

行儀のよいアルゴリズムは、意味のある限りより多くの出力を定義し返さなければなりません。取り立てて特色のない数値や文字列などの出力は、アルゴリズムをより大きなモデルの一部として実行する時に役立ちます。これらの値は、モデル中で続くアルゴリズムの入力パラメータとして使うことができるからです。処理された地物の数や、処理中に見つかった不正な地物の数、出力される地物の数のような数値を、出力に加えることを検討してください。より多くの出力を返せば返すほど、そのアルゴリズムはより役立つものになります。

フィードバック

processAlgorithm() に渡される feedback オブジェクトは、ユーザーからのフィードバックや対話に使用されます。feedback オブジェクトの setProgress() 関数を使うことで、プログレスバー (0 から 100) を更新してアルゴリズムの進捗をユーザに知らせることができます。これはアルゴリズムが完了するまでに時間がかかる場合にとても便利です。

feedback オブジェクトは isCanceled() メソッドを提供し、ユーザによるアルゴリズムのキャンセルを可能にするために監視する必要があります。feedback の pushInfo() メソッドはユーザに情報を送信するために使うことができます。また、reportError() は致命的でないエラーをユーザにプッシュするのに便利です。

アルゴリズムは、ユーザにフィードバックを提供する print 文や QgsMessageLog へのロギングのような、他の形式の使用を避け、常にフィードバックオブジェクトを使うべきです。これはアルゴリズムに対する冗長なロギングを可能にし、スレッドセーフでもあります (これはアルゴリズムが通常バックグラウンドスレッドで実行されることを考えると重要です)。

エラーハンドリング

もしアルゴリズムが無効な入力値や回復できない、あるいは回復できないような状態など、実行を妨げるようなエラーに遭遇した場合、 QgsProcessingException を発生させるべきです。例. :

```
if feature['value'] < 20:
  raise QgsProcessingException('Invalid input value {}, must be >= 20'.format(feature[
↪'value']))
```

QgsProcessingException を致命的でないエラー (例えば、地物のジオメトリが NULL の場合など) に対して発生させないようにし、代わりに feedback.reportError() によってこれらのエラーを報告し、その地物をスキップするようにして下さい。これにより、致命的でないエラーに遭遇した際にアルゴリズム全体の実行を停止する必要がなくなるため、アルゴリズムを「モデルフレンドリ」にすることができます。

スクリプトのドキュメントを作成する

モデルの場合と同様に、スクリプトの追加ドキュメントを作成して、スクリプトの動作や使用方法を説明することができます。

QgsProcessingAlgorithm は、そのために helpString(), shortHelpString() 関数を使います。ユーザにより多くのヘルプを提供するために、これらを指定/オーバーライドして下さい。

shortDescription() は、ツールボックスでアルゴリズムにカーソルを合わせた時のツールチップで使用されます。

26.7.3 実行前後のスクリプトのフック

スクリプトは、アルゴリズムが実行される前と後にそれぞれ実行される実行前フックと実行後フックとしても使用できます。これは、アルゴリズムが実行されるたびに実行されるべきタスクを自動化するために使用できます。

文法は上述のものと同様ですが、 alg という名前のグローバル変数を追加で使用でき、これはたった今 (あるいはまさにこれから) 実行されるアルゴリズムを表します。

プロセシングオプションダイアログの **一般情報** グループには **事前実行スクリプト** と **事後実行スクリプト** という 2 つの項目があり、それぞれのケースで実行するスクリプトのファイル名を入力することができます。

26.8 プロセシングをコマンドラインから使用する

QGIS には QGIS Processing Executor というツールが付属しており、QGIS Desktop 自体を起動することなく、コマンドラインから プロセシングアルゴリズムやモデル（組み込みまたはプラグインで提供）を直接実行することができます。

コマンドラインツールから qgis_process を実行すると、次のようになります：

```
QGIS Processing Executor - 3.27.0-Master 'Master' (3.27.0-Master)
Usage: C:\OSGeo4W\apps\qgis-dev\bin\qgis_process.exe [--help] [--version] [--json] [--
↪verbose] [--no-python] [command] [algorithm id, path to model file, or path to␣
↪Python script] [parameters]

Options:

  --help or -h      Output the help
  --version or -v   Output all versions related to QGIS Process
  --json            Output results as JSON objects
  --verbose         Output verbose logs
  --no-python       Disable Python support (results in faster startup)

Available commands:

  plugins           list available and active plugins
  plugins enable    enables an installed plugin. The plugin name must be specified, e.
↪g. "plugins enable cartography_tools"
  plugins disable   disables an installed plugin. The plugin name must be specified, e.
↪g. "plugins disable cartography_tools"
  list              list all available processing algorithms
  help              show help for an algorithm. The algorithm id or a path to a model␣
↪file must be specified.
  run               runs an algorithm. The algorithm id or a path to a model file and␣
↪parameter values must be specified.
                    Parameter values are specified after -- with PARAMETER=VALUE␣
↪syntax.
                    Ordered list values for a parameter can be created by specifying␣
↪the parameter multiple times,
                    e.g. --LAYERS=layer1.shp --LAYERS=layer2.shp
                    Alternatively, a '-' character in place of the parameters argument␣
↪indicates that the parameters should be read from STDIN as a JSON object.
                    The JSON should be structured as a map containing at least the
↪"inputs" key specifying a map of input parameter values.
                    This implies the --json option for output as a JSON object.
                    If required, the ellipsoid to use for distance and area␣
↪calculations can be specified via the "--ELLIPSOID=name" argument.
                    If required, an existing QGIS project to use during the algorithm␣
↪execution can be specified via the "--PROJECT_PATH=path" argument.
```

注釈: インストールされたプラグインのうち、 metadata.txt ファイルに hasProcessingProvider=yes と記述されているものだけが認識され、qgis_process ツールで起動または読み込まれます。

ヒント: ウィンドウマネージャのないシステム（例 ヘッドレスサーバ）で qgis_process を呼び出す前に、以下を設定する必要があります:

```
export QT_QPA_PLATFORM=offscreen
```

コマンド list を使うと、利用可能なすべてのプロバイダとアルゴリズムのリストを得ることができます。

```
qgis_process list
```

コマンド help を使うと、コマンドやアルゴリズムに関する詳しい情報を得ることができます。

```
qgis_process help qgis:regularpoints
```

コマンド run を使用して、アルゴリズムまたはモデルを実行できます。最初のパラメータとして、アルゴリズムの名前またはモデルへのパスを指定します。

```
qgis_process run native:buffer -- INPUT=source.shp DISTANCE=2 OUTPUT=buffered.shp
```

パラメータが値のリストを受け取る場合、同じ変数を複数回設定します。

```
qgis_process run native:mergevectorlayers -- LAYERS=input1.shp LAYERS=input2.shp␣
↪OUTPUT=merged.shp
```

アルゴリズム実行中はテキスト型のフィードバックバーが表示され、また、 CTRL+C で操作をキャンセルできます。

run コマンドはもっと多くのパラメータをサポートしています。

- --json は、stdout 出力を JSON 構造でフォーマットします。
- --ellipsoid は、指定したものを楕円体に設定します。
- --distance_units は、指定した距離単位を使います。
- --area_units は、指定した面積単位を使います。
- --project_path は、アルゴリズムを実行するために指定したプロジェクトを読み込みます。

複雑な入力パラメータ、すなわちそれ自体がアルゴリズム用の辞書型オブジェクトとして指定されるパラメータ型は、qgis_process によってサポートされています。パラメータが stdin を介して指定されることを示すために、qgis_process コマンドは書式に従わなければなりません（通常の引数リストの代わりに末尾に ``-`` を付けます）。

```
qgis_process run algorithmId -
```

JSON オブジェクトは、入力パラメータ値のマップである "inputs" キーを含んでいなければなりません。例.

```
echo "{'inputs': {'INPUT': 'my_shape.shp', 'DISTANCE': 5}}" | qgis_process run␣
↪native:buffer -
```

さらに、距離単位、面積単位、楕円体、プロジェクトパスなどの追加設定をこの JSON オブジェクトに含めることができます：

```
{
 'ellipsoid': 'EPSG:7019',
 'distance_units': 'feet',
 'area_units': 'ha',
 'project_path': 'C:/temp/my_project.qgs'
 'inputs': {'DISTANCE': 5, 'SEGMENTS': 8 ... }
}
```

入力パラメータを stdin で指定すると、結果の出力形式は自動的に `JSON` になります。

26.9 新たなプロセシングアルゴリズムを Python スクリプトで作成する

Python を使ってプロセシングアルゴリズムを書くためには、ふたつの選択肢があります。

- `QgsProcessingAlgorithm` クラスを **拡張する**
- *@alg* **デコレータを使う**

QGIS 内では、 **プロセシングツールボックス** の上部にある **スクリプト** メニューの **新しいスクリプトを作成** を使用して、自分のコードを書ける **プロセシングスクリプトエディタ** を開いて作成できます。タスクを簡素化するには、同じメニューの **テンプレートから新しいスクリプトを作成する** を使用して、スクリプトテンプレートから開始できます。これにより `QgsProcessingAlgorithm` を拡張するテンプレートが開きます。

スクリプトを `scripts` フォルダ (デフォルトの場所) に `.py` という拡張子で保存するとそのアルゴリズムは **プロセシングツールボックス** で利用できるようになります。

26.9.1 QgsProcessingAlgorithm を拡張する

以下のコードの内容です

1. ベクタレイヤを入力に指定します
2. 地物の数を数えます
3. バッファの操作を行います
4. バッファ操作の結果を使ってラスタレイヤを作ります
5. バッファレイヤ, ラスタレイヤと地物の数を返します

```
from qgis.PyQt.QtCore import QCoreApplication
from qgis.core import (QgsProcessing,
                       QgsProcessingAlgorithm,
                       QgsProcessingException,
                       QgsProcessingOutputNumber,
                       QgsProcessingParameterDistance,
                       QgsProcessingParameterFeatureSource,
                       QgsProcessingParameterVectorDestination,
                       QgsProcessingParameterRasterDestination)
from qgis import processing


class ExampleProcessingAlgorithm(QgsProcessingAlgorithm):
    """
    This is an example algorithm that takes a vector layer,
    creates some new layers and returns some results.
    """

    def tr(self, string):
        """
        Returns a translatable string with the self.tr() function.
        """
        return QCoreApplication.translate('Processing', string)

    def createInstance(self):
        # Must return a new copy of your algorithm.
        return ExampleProcessingAlgorithm()

    def name(self):
        """
        Returns the unique algorithm name.
        """
        return 'bufferrasterextend'

    def displayName(self):
        """
        Returns the translated algorithm name.
        """
        return self.tr('Buffer and export to raster (extend)')

    def group(self):
        """
        Returns the name of the group this algorithm belongs to.
        """
        return self.tr('Example scripts')
```

(次のページに続く)

(前のページからの続き)

```
46
47      def groupId(self):
48          """
49          Returns the unique ID of the group this algorithm belongs
50          to.
51          """
52          return 'examplescripts'
53
54      def shortHelpString(self):
55          """
56          Returns a localised short help string for the algorithm.
57          """
58          return self.tr('Example algorithm short description')
59
60      def initAlgorithm(self, config=None):
61          """
62          Here we define the inputs and outputs of the algorithm.
63          """
64          # 'INPUT' is the recommended name for the main input
65          # parameter.
66          self.addParameter(
67              QgsProcessingParameterFeatureSource(
68                  'INPUT',
69                  self.tr('Input vector layer'),
70                  types=[QgsProcessing.TypeVectorAnyGeometry]
71              )
72          )
73          self.addParameter(
74              QgsProcessingParameterVectorDestination(
75                  'BUFFER_OUTPUT',
76                  self.tr('Buffer output'),
77              )
78          )
79          # 'OUTPUT' is the recommended name for the main output
80          # parameter.
81          self.addParameter(
82              QgsProcessingParameterRasterDestination(
83                  'OUTPUT',
84                  self.tr('Raster output')
85              )
86          )
87          self.addParameter(
88              QgsProcessingParameterDistance(
89                  'BUFFERDIST',
90                  self.tr('BUFFERDIST'),
```

(次のページに続く)

(前のページからの続き)

```
                defaultValue = 1.0,
                # Make distance units match the INPUT layer units:
                parentParameterName='INPUT'
            )
        )
        self.addParameter(
            QgsProcessingParameterDistance(
                'CELLSIZE',
                self.tr('CELLSIZE'),
                defaultValue = 10.0,
                parentParameterName='INPUT'
            )
        )
        self.addOutput(
            QgsProcessingOutputNumber(
                'NUMBEROFFEATURES',
                self.tr('Number of features processed')
            )
        )

    def processAlgorithm(self, parameters, context, feedback):
        """
        Here is where the processing itself takes place.
        """
        # First, we get the count of features from the INPUT layer.
        # This layer is defined as a QgsProcessingParameterFeatureSource
        # parameter, so it is retrieved by calling
        # self.parameterAsSource.
        input_featuresource = self.parameterAsSource(parameters,
                                                     'INPUT',
                                                     context)
        numfeatures = input_featuresource.featureCount()

        # Retrieve the buffer distance and raster cell size numeric
        # values. Since these are numeric values, they are retrieved
        # using self.parameterAsDouble.
        bufferdist = self.parameterAsDouble(parameters, 'BUFFERDIST',
                                            context)
        rastercellsize = self.parameterAsDouble(parameters, 'CELLSIZE',
                                                context)
        if feedback.isCanceled():
            return {}
        buffer_result = processing.run(
            'native:buffer',
            {
```

(次のページに続く)

(前のページからの続き)

```
                # Here we pass on the original parameter values of INPUT
                # and BUFFER_OUTPUT to the buffer algorithm.
                'INPUT': parameters['INPUT'],
                'OUTPUT': parameters['BUFFER_OUTPUT'],
                'DISTANCE': bufferdist,
                'SEGMENTS': 10,
                'DISSOLVE': True,
                'END_CAP_STYLE': 0,
                'JOIN_STYLE': 0,
                'MITER_LIMIT': 10
            },
            # Because the buffer algorithm is being run as a step in
            # another larger algorithm, the is_child_algorithm option
            # should be set to True
            is_child_algorithm=True,
            #
            # It's important to pass on the context and feedback objects to
            # child algorithms, so that they can properly give feedback to
            # users and handle cancelation requests.
            context=context,
            feedback=feedback)

        # Check for cancelation
        if feedback.isCanceled():
            return {}

        # Run the separate rasterization algorithm using the buffer result
        # as an input.
        rasterized_result = processing.run(
            'qgis:rasterize',
            {
                # Here we pass the 'OUTPUT' value from the buffer's result
                # dictionary off to the rasterize child algorithm.
                'LAYER': buffer_result['OUTPUT'],
                'EXTENT': buffer_result['OUTPUT'],
                'MAP_UNITS_PER_PIXEL': rastercellsize,
                # Use the original parameter value.
                'OUTPUT': parameters['OUTPUT']
            },
            is_child_algorithm=True,
            context=context,
            feedback=feedback)

        if feedback.isCanceled():
            return {}
```

(次のページに続く)

(前のページからの続き)

```
        # Return the results
        return {'OUTPUT': rasterized_result['OUTPUT'],
                'BUFFER_OUTPUT': buffer_result['OUTPUT'],
                'NUMBEROFFEATURES': numfeatures}
```

プロセシングアルゴリズムの標準関数:

- **createInstance (必須)**
  自作のアルゴリズムの新しいコピーを返さなければいけません。クラスの名前を変更する場合はここで返す値も必ず一致するように更新して下さい!
- **name (必須)**
  アルゴリズムの識別に使う、ユニークなアルゴリズムの名前を返します。
- **displayName (必須)**
  変換されたアルゴリズム名を返します.
- **group**
  このアルゴリズムが所属しているグループ名を返します.
- **groupId**
  このアルゴリズムが所属しているグループのユニーク ID を返します.
- **shortHelpString**
  このアルゴリズムの翻訳された短縮ヘルプ文字列を返します.
- **initAlgorithm (必須)**
  ここでアルゴリズムの入力と出力を定義します.

  `INPUT` と `OUTPUT` はそれぞれメインの入力と出力パラメータ用の推奨名称です.

  もしパラメータが他のパラメータに依存する場合, `parentParameterName` がこの関係を示します (フィールド/レイヤのバンドまたはレイヤの距離単位は利用できません).
- **processAlgorithm (必須)**
  ここで処理が実行されます.

  パラメータはインスタンスによって `parameterAsSource` や `parameterAsDouble` のような特別な用途の関数を使って取得されます.

  `processing.run` を使うとプロセシングアルゴリズムから他のアルゴリズムを実行できます。最初のパラメータはアルゴリズムの名前です。 2 つめのパラメータはそのアルゴリズム用のパラメータの辞書です。他のアルゴリズム内で実行される場合 `is_child_algorithm` は通常 `True` に設定します。`context` と `feedback` は実行環境とユーザとのやりとりのチャンネルをアルゴリズムに伝えます（キャンセルリクエストのキャッチ、実行状況のレポート、テキストによるフィードバックの提供)。 (親) アルゴリズムのパラメータが "子" アルゴリズムのパラメータとして利用される場合、オリジナルの値を利用する必要があります (例. `parameters['OUTPUT']`)。

  可能な限りキャンセルのためにフィードバックオブジェクトをチェックすることをお勧めします！そうすることで、不要な処理が発生するまでユーザーを待たせずに、すぐにキャンセルすることが可能になります。

アルゴリズムは、辞書として定義したすべての出力パラメータの値を返す必要があります。この場合、それはバッファとラスタライズされた出力レイヤ、および処理された地物数です。辞書キーは、元のパラメータ/出力名と一致する必要があります。

26.9.2 @alg デコレータ

@alg デコレータを使用すると、独自のアルゴリズムを作ることができます。Python コードを書き、それを適切なプロセシングアルゴリズムにする追加情報を提供する数行を加えます。この手法はアルゴリズムの作成、および入力と出力の指定を簡単にします。

アルゴリズム作成にデコレータを使う手法には一つの重要な制約があります．それは作成されたアルゴリズムが常にユーザープロセシングスクリプトプロバイダに追加されることです -- アルゴリズムをカスタムプロバイダに追加して、例えばプラグイン内で使うことはできません．

以下のコードは @alg デコレータを使って次の機能を実装してます

1. ベクタレイヤを入力として利用します
2. 地物の数を数えます
3. バッファ操作を実行します
4. バッファ操作の結果を使ってラスタレイヤを作ります
5. バッファレイヤ, ラスタレイヤと地物の数を返します

```
from qgis import processing
from qgis.processing import alg
from qgis.core import QgsProject

@alg(name='bufferrasteralg', label='Buffer and export to raster (alg)',
     group='examplescripts', group_label='Example scripts')
# 'INPUT' is the recommended name for the main input parameter
@alg.input(type=alg.SOURCE, name='INPUT', label='Input vector layer')
# 'OUTPUT' is the recommended name for the main output parameter
@alg.input(type=alg.RASTER_LAYER_DEST, name='OUTPUT',
           label='Raster output')
@alg.input(type=alg.VECTOR_LAYER_DEST, name='BUFFER_OUTPUT',
           label='Buffer output')
@alg.input(type=alg.DISTANCE, name='BUFFERDIST', label='BUFFER DISTANCE',
           default=1.0)
@alg.input(type=alg.DISTANCE, name='CELLSIZE', label='RASTER CELL SIZE',
           default=10.0)
@alg.output(type=alg.NUMBER, name='NUMBEROFFEATURES',
            label='Number of features processed')

def bufferrasteralg(instance, parameters, context, feedback, inputs):
    """
```

(次のページに続く)

(前のページからの続き)

```
    Description of the algorithm.
    (If there is no comment here, you will get an error)
    """
    input_featuresource = instance.parameterAsSource(parameters,
                                                     'INPUT', context)
    numfeatures = input_featuresource.featureCount()
    bufferdist = instance.parameterAsDouble(parameters, 'BUFFERDIST',
                                            context)
    rastercellsize = instance.parameterAsDouble(parameters, 'CELLSIZE',
                                                context)
    if feedback.isCanceled():
        return {}
    buffer_result = processing.run('native:buffer',
                                {'INPUT': parameters['INPUT'],
                                 'OUTPUT': parameters['BUFFER_OUTPUT'],
                                 'DISTANCE': bufferdist,
                                 'SEGMENTS': 10,
                                 'DISSOLVE': True,
                                 'END_CAP_STYLE': 0,
                                 'JOIN_STYLE': 0,
                                 'MITER_LIMIT': 10
                                 },
                                is_child_algorithm=True,
                                context=context,
                                feedback=feedback)
    if feedback.isCanceled():
        return {}
    rasterized_result = processing.run('qgis:rasterize',
                                {'LAYER': buffer_result['OUTPUT'],
                                 'EXTENT': buffer_result['OUTPUT'],
                                 'MAP_UNITS_PER_PIXEL': rastercellsize,
                                 'OUTPUT': parameters['OUTPUT']
                                },
                                is_child_algorithm=True, context=context,
                                feedback=feedback)
    if feedback.isCanceled():
        return {}
    return {'OUTPUT': rasterized_result['OUTPUT'],
            'BUFFER_OUTPUT': buffer_result['OUTPUT'],
            'NUMBEROFFEATURES': numfeatures}
```

ご覧の通りこのコードには２つのアルゴリズム ('native:buffer' と 'qgis:rasterize') が含まれています. 後の物 ('qgis:rasterize') は最初のもの ('native:buffer') が作成したバッファレイヤを利用してラスタレイヤを作成します.

この処理が行われるコードの部分は、前の章を読んでいれば理解するのは難しくありません。しかし、最

初の行には補足説明が必要です。それらは、あなたのコードをツールボックスやモデルデザイナーなどの GUI コンポーネントから実行できるアルゴリズムにするために必要な情報を提供します。

これらの行は `@alg` デコレータ関数を呼び出しています。その結果アルゴリズムを簡単にコーディングできます。

- @alg デコレータはツールボックス内でのアルゴリズムの名前と場所を定義するのに利用されます。
- @alg.input デコレータはアルゴリズムの入力を定義するのに使われます。
- @alg.output デコレータはアルゴリズムの出力を定義するのに使われます。

26.9.3 プロセシングアルゴリズムのための入力と出力の型

対応する alg デコレータ定数を使用して Processing でサポートされる入力タイプと出力タイプのリストを次に示します（ algfactory.py ファイルには alg 定数の完全なリストが含まれています)。クラス名でソートされています。

入力タイプ

クラス	Alg 定数	説明
QgsProcessingParameterAnnotationLayer	alg.ANNOTATION_LAYER	注記レイヤ
QgsProcessingParameterAuthConfig	alg.AUTH_CFG	ユーザに利用可能な認証構成を選択させるか新しい認証構成を作成させます
QgsProcessingParameterBand	alg.BAND	ラスタレイヤのバンド
QgsProcessingParameterBoolean	alg.BOOL	ブール値
QgsProcessingParameterColor	alg.COLOR	色
QgsProcessingParameterCoordinateOperat	alg.COORDINATE_OPERATION	座標操作（CRS 変換）
QgsProcessingParameterCrs	alg.CRS	座標参照系
QgsProcessingParameterDatabaseSchema	alg.DATABASE_SCHEMA	データベーススキーマ
QgsProcessingParameterDatabaseTable	alg.DATABASE_TABLE	データベーステーブル
QgsProcessingParameterDateTime	alg.DATETIME	日付時刻（または日付または時刻）
QgsProcessingParameterDistance	alg.DISTANCE	距離の値用の倍精度数値パラメータ
QgsProcessingParameterEnum	alg.ENUM	事前定義された複数の値からの選択を許す列挙
QgsProcessingParameterExpression	alg.EXPRESSION	式
QgsProcessingParameterExtent	alg.EXTENT	xmin, xmax, ymin, ymax で定義された空間領域

次のページに続く

表 26.3 – 前のページからの続き

クラス	Alg 定数	説明
QgsProcessingParameterField	alg.FIELD	ベクタレイヤの属性テーブルのフィールド
QgsProcessingParameterFile	alg.FILE	既存ファイルのファイル名
QgsProcessingParameterFileDestination	alg.FILE_DEST	新たに作成された出力ファイルのファイル名
QgsProcessingParameterFolderDestinatic	alg.FOLDER_DEST	フォルダー(保存先フォルダー)
QgsProcessingParameterGeometry	alg.GEOMETRY	ジオメトリ
QgsProcessingParameterNumber	alg.INT	整数
QgsProcessingParameterLayout	alg.LAYOUT	レイアウト
QgsProcessingParameterLayoutItem	alg.LAYOUT_ITEM	レイアウトアイテム
QgsProcessingParameterMapLayer	alg.MAPLAYER	マップレイヤ
QgsProcessingParameterMapTheme	alg.MAP_THEME	プロジェクトマップテーマ
QgsProcessingParameterMatrix	alg.MATRIX	配列
QgsProcessingParameterMeshLayer	alg.MESH_LAYER	メッシュレイヤ
QgsProcessingParameterMultipleLayers	alg.MULTILAYER	レイヤのセット
QgsProcessingParameterNumber	alg.NUMBER	数値
QgsProcessingParameterPoint	alg.POINT	点
QgsProcessingParameterPointCloudDestir	alg. POINTCLOUD_LAYER_DEST	アルゴリズムによって作成される点群レイヤの保存先パスを指定する、点群レイヤ保存先パラメータ
QgsProcessingParameterPointCloudLayer	alg.POINTCLOUD_LAYER	点群レイヤ
QgsProcessingParameterProviderConnecti	alg.PROVIDER_CONNECTION	データベースプロバイダの利用可能な接続
QgsProcessingParameterRange	alg.RANGE	数値の範囲
QgsProcessingParameterRasterLayer	alg.RASTER_LAYER	ラスタレイヤ
QgsProcessingParameterRasterDestinatic	alg.RASTER_LAYER_DEST	ラスタレイヤの保存先パラメータ、アルゴリズムが作成したラスタレイヤの保存先パスを指定するためのパラメータ
QgsProcessingParameterScale	alg.SCALE	地図の縮尺
QgsProcessingParameterFeatureSink	alg.SINK	地物シンク
QgsProcessingParameterFeatureSource	alg.SOURCE	地物ソース
QgsProcessingParameterString	alg.STRING	テキストストリング
QgsProcessingParameterVectorLayer	alg.VECTOR_LAYER	ベクターレイヤー

次のページに続く

表 26.3 – 前のページからの続き

クラス	Alg 定数	説明
QgsProcessingParameterVectorDestinatic	alg.VECTOR_LAYER_DEST	アルゴリズムで作成されたベクタレイヤの保存先パスを指定するためのベクタレイヤの保存先パラメータ

出力タイプ

クラス	Alg 定数	説明
QgsProcessingOutputBoolean	alg.BOOL	ブール値
QgsProcessingOutputNumber	alg.DISTANCE	距離の値用の倍精度数値パラメータ
QgsProcessingOutputFile	alg.FILE	既存ファイルのファイル名
QgsProcessingOutputFolder	alg.FOLDER	フォルダ
QgsProcessingOutputHtml	alg.HTML	HTML
QgsProcessingOutputNumber	alg.INT	整数
QgsProcessingOutputLayerDefinition	alg.LAYERDEF	レイヤ定義
QgsProcessingOutputMapLayer	alg.MAPLAYER	マップレイヤ
QgsProcessingOutputMultipleLayers	alg.MULTILAYER	レイヤのセット
QgsProcessingOutputNumber	alg.NUMBER	数値
QgsProcessingOutputPointCloudLayer	alg.POINTCLOUD_LAYER	点群レイヤ
QgsProcessingOutputRasterLayer	alg.RASTER_LAYER	ラスタレイヤ
QgsProcessingOutputString	alg.STRING	テキストストリング
QgsProcessingOutputVectorLayer	alg.VECTOR_LAYER	ベクターレイヤー

26.9.4 アルゴリズムの出力を渡す

出力としてレイヤ（ラスタまたはベクタ）を指定すると、アルゴリズムは終了後にそれを QGIS に追加しようとします。

- ラスタレイヤ出力: QgsProcessingParameterRasterDestination / alg.RASTER_LAYER_DEST.
- ベクタレイヤ出力: QgsProcessingParameterVectorDestination / alg.VECTOR_LAYER_DEST.

したがって `processing.run()` メソッドが作成したレイヤをユーザの現在のプロジェクトに加えない場合でも、２つの出力レイヤ (バッファとラスタバッファ) が読み込まれます．これらはユーザが入力した保存先 (またはユーザが保存先を指定しない場合は一時的な保存先) に保存されるからです．

レイヤがアルゴリズムの出力として作成される場合、そのように宣言する必要があります。そうしないと、宣言されたものがアルゴリズムが実際に作成するものと一致しないため、モデラーでアルゴリズムを適切に使用できません。

結果の辞書で文字列、数値などを指定することで返すことができます（「NUMBEROFFEATURES」で示したとおり）が、それらは常にアルゴリズムからの出力として明示的に定義する必要があります。アルゴリズムがモデルの一部として使用される場合、これらのアルゴリズムは後のアルゴリズムで使用するのに役立つ可能性があるため、アルゴリズムではできるだけ多くの有用な値を出力することをを推奨します。

26.9.5 ユーザーとやりとりする

あなたのアルゴリズムが処理に長い時間を必要とする場合、ユーザに進捗状況を知らせるのは良い考えです. `feedback` (`QgsProcessingFeedback`) を利用するとそれを実現できます.

処理状況テキストとプログレスバーは２つのメソッドを使って更新できます: `setProgressText(text)` と `setProgress(percent)` です.

次のメソッドを使うとさらに多くの情報を提供できます `pushCommandInfo(text)`, `pushDebugInfo(text)`, `pushInfo(text)` と `reportError(text)`.

スクリプトに問題がある場合、それを処理する正しい方法は `QgsProcessingException` を発生させることです。メッセージを引数として例外のコンストラクタに渡すことができます。プロセシングでは、アルゴリズムの実行元（ツールボックス、モデラー、Python コンソールなど）に応じてプロセシングとユーザーとの通信を処理します。

26.9.6 スクリプトのドキュメントを作成する

以下のものをオーバーロードするとあなたのスクリプトのドキュメントを作れます `helpString()` と `helpUrl()` methods of `QgsProcessingAlgorithm`.

26.9.7 フラグ

`QgsProcessingAlgorithm` の `flags()` メソッドをオーバーライドすると、QGIS にアルゴリズムについてより詳しく伝えることができます。例えば、スクリプトをモデラーから隠すこと、キャンセルすること、スレッドセーフでないこと、などを QGIS に伝えることができます。

Tip: デフォルトでは、処理タスクの実行中に QGIS の応答性を維持するために、Processing はアルゴリズムを別のスレッドで実行します。アルゴリズムが定期的にクラッシュする場合は、おそらくバックグラウンドスレッドで安全に実行できない API 呼び出しを使用している可能性があります。アルゴリズムの flags() メソッドから QgsProcessingAlgorithm.FlagNoThreading フラグを返して、代わりにメインスレッドでアルゴリズムを実行するように強制します。

26.9.8 スクリプトアルゴリズムを書くためのベストプラクティス

スクリプトアルゴリズムを作成する際、特に他の QGIS ユーザーと共有したい場合に考慮すべきアイデアの簡単な概要を以下に示します。これらの単純な規則に従うことで、ツールボックス、モデラー、バッチ処理インターフェースなどのさまざまな処理要素間で一貫性が確保されます。

- 結果のレイヤは読み込まないでください。結果はプロセシングに処理させ、必要であればレイヤを読み込ませてください。
- あなたのアルゴリズム作成する出力について常に宣言して下さい.
- メッセージボックスを表示したり、スクリプトの GUI 要素を使用したりしないでください。ユーザーと通信する場合は、フィードバックオブジェクトのメソッド（`QgsProcessingFeedback`）を使用するか、`QgsProcessingException` をスローします。

QGIS にはすでに多くのプロセシングアルゴリズムが用意されています。コードは QGIS のレポにあります。

26.10 外部アプリケーションの設定

プロセシングフレームワークは、追加のアプリケーションを使って拡張できます。外部アプリケーションに依存するアルゴリズムは、独自のアルゴリズムプロバイダによって管理されます。追加のプロバイダは個別のプラグインとして見つけることができ、QGIS プラグインマネージャを使ってインストールできます。

このセクションでは、これらの追加アプリケーションを含むようにプロセシングフレームワークを設定する方法を示し、それらに基づいたアルゴリズムのいくつかの特別な機能について説明します。一度システムを正しく設定すれば、他のアルゴリズムと同じように、ツールボックスやモデルデザイナーなど、どのコンポーネントからでも外部アルゴリズムを実行することができるようになります。

デフォルトでは、QGIS に同梱されていない外部アプリケーションに依存するアルゴリズムは有効になっていません。システムにインストールされている場合は、プロセシング設定ダイアログで有効にすることができます。

26.10.1 Windows ユーザーへの注意点

あなたが上級ユーザーでなく、Windows 上で QGIS を実行している場合、この章の残りを読むことに興味がないかもしれません。スタンドアロンインストーラを使って QGIS をあなたのシステムにインストールしてください。これにより、SAGA と GRASS が自動的にインストールされ、QGIS から実行できるように設定されます。これらのプロバイダから提供されるすべてのアルゴリズムは、それ以上の設定を必要とすることなく実行できるようになります。OSGeo4W アプリケーションを使ってインストールする場合は、SAGA と GRASS を選択してインストールするようにしてください。

26.10.2 ファイル形式に関する注意点

外部ソフトウェアを使用する場合、QGIS でファイルを開いたからといって、そのファイルをその外部ソフトウェアで開いて処理できるわけではありません。ほとんどの場合、QGIS で開いたファイルを他のソフトウェアで読み込むことができますが、そうでない場合もあります。ラスタレイヤ、ベクタレイヤを問わず、データベースや一般的でないファイル形式を使用する場合、問題が発生する可能性があります。そのような場合は、両方のプログラムが確実に理解できる、よく知られたファイル形式を使用するようにし、コンソール出力（ログパネル）をチェックして、何が間違っているのかを調べてください。

例えば、GRASS のラスタレイヤを入力として外部のアルゴリズムを呼び出すと、トラブルが発生し、作業が完了しない可能性があります。このため、そのようなレイヤはアルゴリズムに利用可能なレイヤとして表示されません。

しかしながら、ベクタレイヤについては問題が起きません。QGIS はベクタレイヤを外部アプリケーションに渡す前に、元のファイル形式から外部アプリケーションで受け入れられる形式に自動的に変換するためです。これには余分な処理時間が加わり、大きなレイヤで顕著でしょう。ですから、同じようなサイズの Shapefile 形式のデータセットからのレイヤよりも、DB 接続からのレイヤの処理に時間がかかっても驚かないでください。

外部アプリケーションを使用していないプロバイダは QGIS で開けるレイヤはどれも処理できます。なぜならそれらは QGIS を通じて分析のためにそれを開いているので。

QGIS で生成されるすべてのラスタおよびベクタ出力形式は、入力レイヤとして使用できます。一部のプロバイダは特定の形式をサポートしていませんが、すべてのプロバイダは、後に QGIS が自動的に変換できる一般的な形式にエクスポートできます。入力レイヤについては、変換が必要な場合、処理時間が長くなる可能性があります。

26.10.3 ベクタレイヤの選択に関する注意点

外部アプリケーションが QGIS 内でベクタレイヤに存在する選択を知らされることもできます。しかし、それにはそれらが元から外部アプリケーションでサポートされていない形式であったかのように、すべての入力ベクタレイヤを書き換えることが必要です。選択が存在しないか、 **選択された地物のみを使用** オプションがプロセシングの一般設定で有効になっていない場合にのみ、レイヤは外部アプリケーションに直接渡すことができます。

また、選択した地物だけをエクスポートする必要がある場合もあり、その場合は実行時間が長くなります。

26.10.4 SAGA: System for Automated Geoscientific Analyses、地球科学自動分析システム

SAGA が QGIS のインストールに含まれている場合、SAGA アルゴリズムを QGIS から実行することができます。

Windows を使っているなら、スタンドアロンインストーラーと OSGeo4W インストーラーの両方に SAGA が含まれています。

SAGA グリッドシステムの制限について

複数の入力ラスターレイヤーを必要とするほとんどの SAGA アルゴリズムは、それらが同じグリッド系を持つことを必要としています。つまり、それらが同じ地理的領域をカバーして同じセルサイズを持ち、対応するグリッドが一致する必要があります。QGIS から SAGA アルゴリズムを呼び出すときは、そのセルサイズと範囲に関係なく、どんなレイヤーも使用できます。複数のラスターレイヤーは SAGA アルゴリズムのための入力として使用される場合、QGIS は（SAGA アルゴリズムが異なるグリッド系からのレイヤーで動作できない場合）共通のグリッド系にそれらをリサンプリングした後、SAGA に渡します。

その一般的なグリッド系の定義は、ユーザーによって制御され、そうする設定ウィンドウの SAGA グループ内のいくつかのパラメーターがあります。ターゲットグリッド系を設定する 2 つの方法があります。

- 手動で設定する。次のパラメーターの値を設定することによって範囲を定義します。
  - **リサンプリング最小** *X*
  - **リサンプリング最大** *X*
  - **リサンプリング最小** *Y*
  - **リサンプリング最大** *Y*
  - **リサンプリングセルサイズ**

  QGIS はその範囲に対して入力レイヤーを、それらがその範囲と重複していない場合でも、再サンプリングすることに注意してください。
- 入力レイヤーから自動的に設定する。このオプションを選択するには、 **リサンプリングにグリッド系の最小カバーを使用** オプションチェックするだけです。他のすべての設定は無視され、すべての入力レイヤーを最小でカバーする範囲が使用されます。ターゲットレイヤーのセルサイズは、入力レイヤーの全てのセルサイズの最大値です。

多重ラスターレイヤーを使用しないあるいは固有の入力グリッドシステムを必要としないアルゴリズムでは、SAGA を呼び出す前にリサンプリングは実行されませんし、これらのパラメーターは使用されません。

マルチバンドレイヤーに関する制限

QGIS とは異なり、SAGA ではマルチバンドレイヤーをサポートしていません。（例えば RGB またはマルチスペクトル画像のような）マルチバンドレイヤーを使用する場合は、最初にシングルバンド化された画像に分割する必要があります。そうするためには、「SAGA/グリッドツール/RGB 画像を分割」アルゴリズム（RGB 画像から三つの画像を作成する）や「SAGA /グリッドツール/バンド抽出」アルゴリズム（単一バンドを抽出する）を使用できます。

セルサイズの制限

SAGA ではラスターレイヤーが x 軸と y 軸において同じセルサイズであることを仮定しています。もし、水平方向と垂直方向でセルサイズが異なる値のレイヤーで作業するならば、予期せぬ結果を得ることになるでしょう。この場合、入力レイヤーが SAGA によって適切に処理されないであろうという警告がプロセスログに加えられることになります。

ログを記録する

QGIS は SAGA を呼び出すと、それはこれにより、すべての必要な操作を実行するためにコマンドのセットを通過する、そのコマンドラインインターフェイスを使用しません。SAGA は、追加のコンテンツと共に、既に行われる処理のパーセンテージを含むコンソールに情報を書き込むことにより、その進捗状況を示しています。この出力はフィルタがかけられ、アルゴリズムの実行中に、プログレスバーを更新するために使用されます。

QGIS によって送信されたコマンドと SAGA によって出力された追加情報は、他のプロセシングログメッセージと一緒にログに記録することができ、QGIS が SAGA アルゴリズムを実行するときに何が起こっているかを追跡するのに便利です。このロギングメカニズムを有効にするための 2 つの設定、すなわち **コンソール出力をログに記録する** と **実行コマンドをログに記録する** があります。

外部アプリケーションを使用し、コマンドラインから呼び出す他のほとんどのプロバイダにも同様のオプションがあるため、プロセシング設定リストの他の場所にも同様のオプションがあります。

26.10.5 R スクリプト

プロセシングで R を有効にするには、**Processing R Provider** プラグインをインストールして QGIS 用に R を設定する必要があります。

設定は **設定** ▸ **オプション** の **プロセシング** タブにある **プロバイダ** ▸ *R* で行ないます。

オペレーティング・システムによっては、 *R folder* を使って R バイナリの場所を指定する必要があるかもしれません。

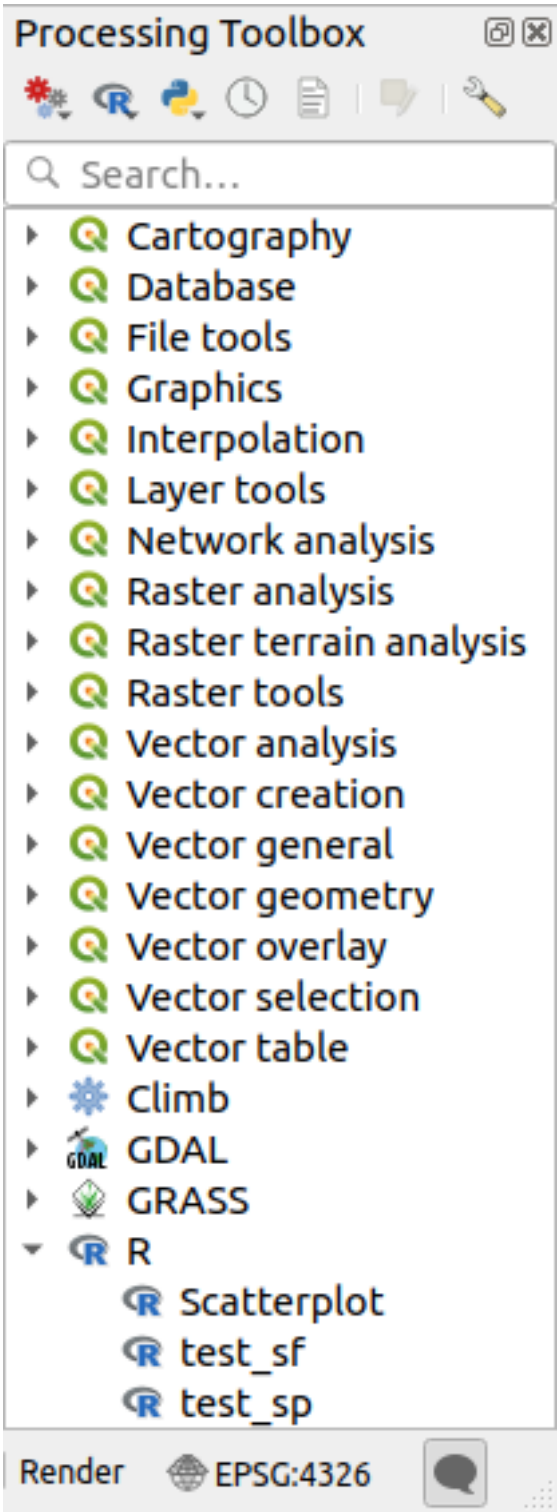
注釈: **Windows** の場合、R の実行ファイルは通常 `C:\Program Files\R\` の下のフォルダ (R-<version>) にあります。バイナリ **ではなく** フォルダを指定してください！

Linux では、R フォルダが PATH 環境変数に入っていることを確認するだけです。ターミナル・ウィンドウで R が R を起動すれば、準備は完了です。

Processing R Provider プラグインをインストールすると、いくつかのサンプルスクリプトが **プロセシングツールボックス** に見つかります：

- *Scatterplot* は与えられたベクタレイヤの 2 つの数値フィールドから散布図を作成する R 関数を実行します。

- *test_sf* は `sf` パッケージに依存するいくつかの操作を行い、R パッケージ `sf` がインストールされているかどうかをチェックするために使うことができます。パッケージがインストールされていない場合、R はプロセシングオプションの **プロバイダ** → *R* で指定された *Package repository* を使って、パッケージ（および依存するすべてのパッケージ）のインストールを試みます。デフォルトは https://cran.r-project.org/ です。インストールには時間がかかるかもしれません…
- *test_sp* を使うと、R パッケージ `sp` がインストールされているかどうかをチェックできます。このパッケージがインストールされていない場合、R がインストールを試みます。



R が QGIS 用に正しく設定されていれば、これらのスクリプトを実行できるはずです。

QGIS コレクションから R スクリプトを追加する

QGIS における R の統合は SAGA とは異なり、あらかじめ定義された実行可能なアルゴリズムのセットはありません（*Processing R Provider* プラグインに付属するいくつかのスクリプト例を除く）。

サンプル R スクリプトのセットは QGIS Repository で利用可能です。以下の手順を実行し、*QGIS Resource Sharing* プラグインを使用してそれらを読み込んで有効にします。

1. *QGIS Resource Sharing* プラグインを追加します（プラグインマネージャの **設定** で **実験的プラグインも表示** を有効にする必要があるかもしれません）。
2. それを開きます（プラグイン --> Resource Sharing --> Resource Sharing）
3. *Settings* タブを選びます
4. *Reload repositories* をクリックします
5. *All* タブを選びます
6. リストから *QGIS R script collection* を選び、*Install* ボタンをクリックします
7. これでそのコレクションが *Installed* タブにリストされているはずです
8. プラグインを閉じます
9. **プロセシングツールボックス** を開きます。問題がなければ、スクリプトの例が R の下に様々なグループで表示されます（下のスクリーンショットでは一部のグループのみが展開されています）。

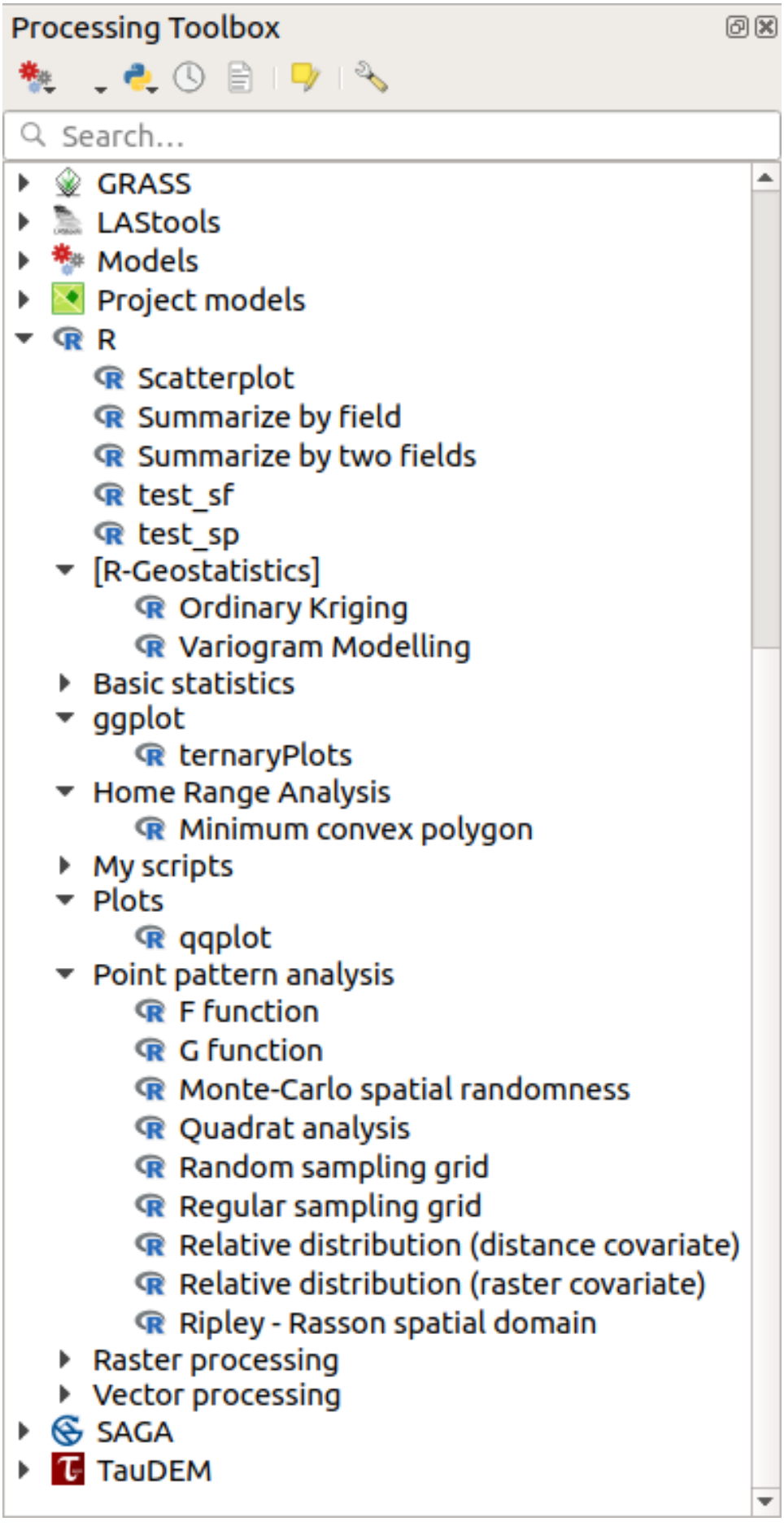


図 26.34: いくつかの R スクリプトが表示された **プロセシングツールボックス**

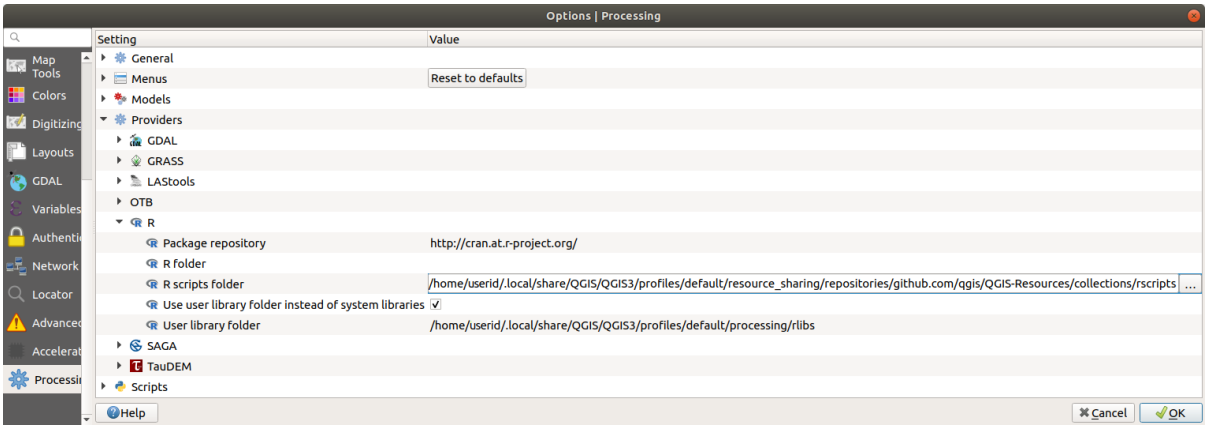
一番上のスクリプトは、*Processing R Provider* プラグインのスクリプト例です。

10. もし何らかの理由でスクリプトが **プロセシングツールボックス** で利用できない場合は、次を試してみてください：

    1. プロセシング設定を開く（**設定** ▸ **オプション** ▸ **プロセシング** タブ）
    2. **プロバイダ** ▸ *R* ▸ *R scripts folder* に移動する

       - Ubuntu ではそのパスを次に設定する（またはそのパスに含めた方が良い）：

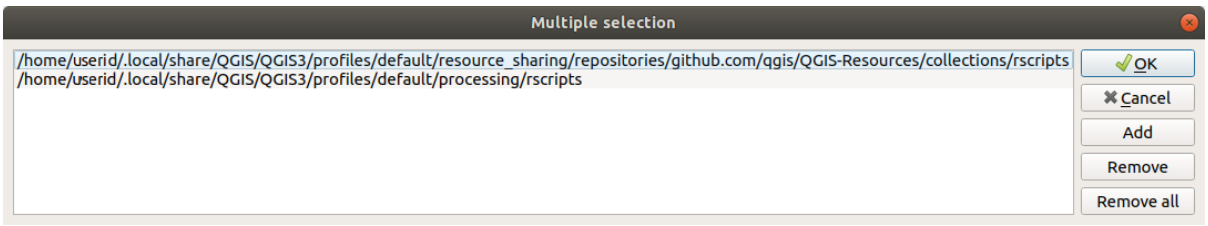
         /home/<user>/.local/share/QGIS/QGIS3/profiles/default/resource_sharing/repositories/github.com/qgis/QGIS-Resources/collections/rscripts



       - Windows ではそのパスを次に設定する（またはそのパスに含めた方が良い）：

         C:\Users\<user>\AppData\Roaming\QGIS\QGIS3\profiles\default\resource_sharing\repositories\github.com\qgis\QGIS-Resources\collections\rscripts

    編集するにはダブルクリックします。その後、パスを貼り付けたり入力したりするか、... ボタンでディレクトリに移動し、開いたダイアログで **追加** ボタンを押します。ここでは複数のディレクトリを指定することができます。それらはセミコロン (";") で区切られます。



QGIS 2 オンラインコレクションからすべての R スクリプトを取得したい場合は、*QGIS R script collection* の代わりに *QGIS R script collection (from QGIS 2)* を選択してください。おそらく、ベクタデータの入出力に依存するスクリプトは動作しないことがわかります。

R スクリプトを作る

R から行うのと同じように、スクリプトを書いたり、R コマンドを呼び出したりすることができます。このセクションでは、QGIS で R コマンドを使用するための構文と、その中で QGIS オブジェクト（レイヤ、テーブル）を利用する方法について説明します。

R 関数を呼び出すアルゴリズム（または、開発したより複雑な R スクリプトを QGIS から利用できるようにしたい場合）を追加するには、R コマンドを実行するスクリプトファイルを作成する必要があります。

R スクリプトファイルの拡張子は `.rsx` で、R 構文と R スクリプトの基本的な知識があれば、作成はとても簡単です。これらのファイルは R スクリプトフォルダに保存する必要があります。このフォルダ（*R scripts folder*）は、プロセシング設定ダイアログの *R* 設定グループで指定することができます）。

R のメソッド `spsample` を呼び出して、指定したポリゴンレイヤのポリゴンの境界内にランダムなグリッドを作成する、非常にシンプルなスクリプトファイルを見てみましょう。このメソッドは `maptools` パッケージに属しています。QGIS に組み込みたいアルゴリズムのほとんどが空間データを使用または生成するため、`maptools` や `sp/sf` のような空間パッケージの知識は非常に役に立ちます。

```
##Random points within layer extent=name
##Point pattern analysis=group
##Vector_layer=vector
##Number_of_points=number 10
##Output=output vector
library(sp)
spatpoly = as(Vector_layer, "Spatial")
pts=spsample(spatpoly,Number_of_points,type="random")
spdf=SpatialPointsDataFrame(pts, as.data.frame(pts))
Output=st_as_sf(spdf)
```

Python の二重のコメント記号（##）で始まる最初の行は、スクリプトの表示名とグループを定義し、QGIS に入力と出力を伝えます。

注釈: 自分で R スクリプトを書く方法を詳しく知るには、トレーニングマニュアルの R Intro セクションを参照し、 *QGIS R Syntax* セクションを参照してください。

入力パラメータを宣言すると、QGIS はその情報を 2 つのことに使用します：そのパラメータの値をユーザに尋ねるユーザインタフェースを作成することと、R 関数の入力として使用できる対応した R 変数を作成することです。

上記の例では、`Vector_layer` という `vector` 型の入力を宣言しました。そのアルゴリズムを実行するとき、QGIS はユーザが選択したレイヤを開き、それを `Vector_layer` という変数に格納します。つまり、パラメータ名はそのパラメータの値にアクセスするために使用する R の変数名となります（そのため、R の予約語をパラメータ名として使用することは避けるべきです）。

ベクタレイヤやラスタレイヤなどの空間パラメータは `st_read()` (または `readOGR`) や `brick()` (または `readGDAL`) コマンドを使って読み込まれ（これらのコマンドを記述ファイルに追加する必要はありません -- QGIS が行います）、`sf` (または `Spatial*DataFrame`) オブジェクトとして保存されます。

テーブルフィールドは、選択されたフィールド名を格納する文字列として保存されます。

ベクタファイルは、##load_vector_using_rgdal を指定すると、 st_read() の代わりに readOGR() コマンドを使って読み込むことができます。これは sf オブジェクトの代わりに Spatial*DataFrame オブジェクトを生成します。

ラスタファイルは、##load_raster_using_rgdal を指定すると、brick() の代わりに readGDAL() コマンドを使って読み込むことができます。

もしあなたが上級ユーザーで、QGIS にレイヤのオブジェクトを作成させたくない場合、##pass_filenames を使用することで、ファイル名を文字列で指定することができます。この場合、そのファイルに含まれるデータに対して操作を行う前に、ファイルを開く必要があります。

以上の情報で、R スクリプトの最初の行（Python のコメント文字で始まっていない最初の行）を理解することができます。

```
library(sp)
spatpoly = as(Vector_layer, "Spatial")
pts=spsample(polyg,numpoints,type="random")
```

spsample 関数は *sp* ライブラリで提供されているので、最初にすることはそのライブラリを読み込むことです。変数 Vector_layer には sf オブジェクトが格納されています。ここでは *sp* ライブラリの関数（spsample）を使用するので、as 関数を使用して sf オブジェクトを SpatialPolygonsDataFrame オブジェクトに変換する必要があります。

次に、このオブジェクトと numpoints 入力パラメータ（生成するポイントの数を指定します）を使用して spsample 関数を呼び出します。

Output という名前のベクタ出力を宣言したので、sf オブジェクトを含む Output という名前の変数を作成しなければなりません。

これを 2 つのステップで行います。まず、*SpatialPointsDataFrame* 関数を使用して、関数の結果から SpatialPolygonsDataFrame オブジェクトを作成し、次に、(*sf* ライブラリの) st_as_sf 関数を使用して、そのオブジェクトを sf オブジェクトに変換します。

中間変数には好きな名前を使うことができます。ただ、最終的な結果を格納する変数が定義された名前（この場合は Output）を持ち、適切な値（ベクタレイヤの出力には sf オブジェクト）を含んでいることを確認してください。

この場合、spsample メソッドから得られた結果は、SpatialPointsDataFrame オブジェクトを介して、明示的に sf オブジェクトに変換する必要がありました。なぜなら、それ自体が ppp クラスのオブジェクトであり、QGIS に返すことができないからです。

アルゴリズムは、ラスタレイヤを生成した場合、それらが保存されている方法は、##dontuserasterpackage オプションを使用しているかどうかに依存します。それを使用している場合は、これらのレイヤは、writeGDAL() メソッドを使用して保存されます。そうでない場合、raster パッケージから writeRaster() メソッドが使用されます。

##pass_filenames オプションを使用した場合、出力は raster パッケージを使用して生成されます（ writeRaster() を使用）。

アルゴリズムがレイヤを生成せず、代わりにコンソールにテキスト結果を表示する場合、実行が終了したらコンソールに表示させたいことを示す必要があります。そのためには、表示させたい結果を生成するコマンド行を > （「大なり」）記号で始めるだけです。行頭に > が付いた行の出力のみが表示されます。例え

ば、ベクタレイヤの属性の指定されたフィールド（カラム）に対して正規性テストを実行するアルゴリズムの記述ファイルを以下に示します：

```
##layer=vector
##field=field layer
##nortest=group
library(nortest)
>lillie.test(layer[[field]])
```

最後の行の出力が印刷されているが、第一の出力はない（そしていずれも、他のコマンドラインからの出力は、QGIS によって自動的に追加されています）。

もしあなたのアルゴリズムが（`plot()``メソッドを使って）何らかのグラフィックを作成するのであれば、以下の行を追加してください（``output_plots_to_html` は以前は `showplots` でした）：

```
##output_plots_to_html
```

これは、QGIS は R の実行が完了した後に開かれる一時ファイルにすべての R のグラフィック出力をリダイレクトするようになります。

グラフィックスとコンソールの両方の結果は、プロセシング結果マネージャから入手できます。

詳しくは、QGIS の公式コレクションにある R スクリプトを見てください（別の場所で説明されているように、*QGIS Resource Sharing* プラグインを使ってダウンロードし、インストールします）。ほとんどのスクリプトは非常にシンプルで、独自のスクリプトを作成する方法を理解するのに非常に役立ちます。

注釈: `sf`、`rgdal`、`raster` ライブラリはデフォルトで読み込まれるので、対応する `library()` コマンドを追加する必要はありません。しかし、その他の必要なライブラリは明示的に読み込む必要があります：library(ggplot2)`` （`ggplot2` ライブラリを読み込む）。もしそのパッケージがあなたのマシンにまだインストールされていない場合、プロセシングはそのパッケージをダウンロードしてインストールしようとします。こうすることで、R Standalone でもパッケージを利用できるようになります。パッケージのダウンロードが必要な場合、スクリプトの初回実行に時間がかかる可能性があることに **注意してください**。

26.10.6 R ライブラリ

R スクリプト sp_test は、R パッケージ sp と raster を読み込もうとします。

sf_test の実行時にインストールされる R ライブラリ

R スクリプト *sf_test* は、sf と raster を読み込もうとします。これら二つのパッケージがインストールされていないとき、R はそれら（とそれが依存する全てのライブラリ）を読み込んでインストールしようとするでしょう。

以下の R ライブラリは、Ubuntu の Processing Toolbox から *Processing R Provider* プラグインのバージョン 2.0 と *R* 3.4.4（*apt* パッケージの r-base-core のみ）を新規インストールして sf_test を実行した後、~/.local/share/QGIS/QGIS3/profiles/default/processing/rscripts` に格納されます：

```
abind, askpass, assertthat, backports, base64enc, BH, bit, bit64, blob, brew, callr,␣
↪classInt, cli, colorspace, covr, crayon, crosstalk, curl, DBI, deldir,
desc, dichromat, digest, dplyr, e1071, ellipsis, evaluate, fansi, farver, fastmap,␣
↪gdtools, ggplot2, glue, goftest, gridExtra, gtable, highr, hms,
htmltools, htmlwidgets, httpuv, httr, jsonlite, knitr, labeling, later, lazyeval,␣
↪leafem, leaflet, leaflet.providers, leafpop, leafsync, lifecycle, lwgeom,
magrittr, maps, mapview, markdown, memoise, microbenchmark, mime, munsell, odbc,␣
↪openssl, pillar, pkgbuild, pkgconfig, pkgload, plogr, plyr, png, polyclip,
praise, prettyunits, processx, promises, ps, purrr, R6, raster, RColorBrewer, Rcpp,␣
↪reshape2, rex, rgeos, rlang, rmarkdown, RPostgres, RPostgreSQL,
rprojroot, RSQLite, rstudioapi, satellite, scales, sf, shiny, sourcetools, sp,␣
↪spatstat, spatstat.data, spatstat.utils, stars, stringi, stringr, svglite,
sys, systemfonts, tensor, testthat, tibble, tidyselect, tinytex, units, utf8, uuid,␣
↪vctrs, viridis, viridisLite, webshot, withr, xfun, XML, xtable
```

26.10.7 GRASS: Geographic Resources Analysis Support System、地理的資源分析支援システム

GRASS を設定するのは SAGA を設定するのとあまり変わりません。Windows をお使いの場合は、まず GRASS フォルダのパスを定義する必要があります。

デフォルトでは、プロセシングフレームワークは、QGIS に同梱されている GRASS ディストリビューションを使用するように GRASS コネクタを設定しようとします。ほとんどのシステムでは問題なく動作するはずですが、問題が発生した場合は、GRASS コネクタを手動で設定する必要があるかもしれません。また、別の GRASS を使用したい場合は、別のバージョンがインストールされているフォルダを指すように設定を変更することができます。アルゴリズムを正しく動作させるには GRASS 7 が必要です。

Linux をお使いの場合は、GRASS が正しくインストールされ、ターミナルウィンドウから問題なく実行できることを確認してください。

GRASS アルゴリズムは、計算するための領域を使用します。この領域は、アルゴリズムを毎回実行するために使用されるすべての入力レイヤーを覆う最小範囲をとる、自動的 SAGA 構成に見られるものと同様の

値を用いて、または手動で定義できます。後者のアプローチが好む動作である場合、GRASS の設定パラメーターで **最小カバー領域を使用** オプションをチェックするだけです。

26.10.8 LAStools

QGIS で LAStools を使うには、LAStools をコンピュータにダウンロードしてインストールし、LAStools プラグイン（公式リポジトリから入手可能です）を QGIS にインストールする必要があります。

Linux プラットフォームでは、いくつかのツールを実行するために Wine が必要です。

LAStools は Processing options (**設定** **オプション**, **プロセシング** タブ, **プロバイダ** *LAStools*) で有効化され、設定されます。ここで LAStools (*LAStools* **フォルダ**) と Wine (*Wine* **フォルダ**) の場所を指定できます。Ubuntu の場合、デフォルトの Wine フォルダは `/usr/bin` です。

26.10.9 OTB アプリケーション

OTB アプリケーションは QGIS プロセシングフレームワークで完全にサポートされています。

OTB (Orfeo ToolBox) は、リモートセンシングデータ用の画像処理ライブラリです。また、画像処理機能を提供するアプリケーションも提供されています。アプリケーションの一覧とそのドキュメントは OTB CookBook に掲載されています

注釈: OTB は QGIS と一緒に配布されていないので、別途インストールする必要があることに注意してください。OTB のバイナリパッケージは download page にあります。

OTB ライブラリを見つけるように QGIS プロセシングを設定するには：

1. プロセシングの設定を開きます: **設定** **オプション** **プロセシング** （左のパネル）*
2. "プロバイダ"の下に OTB が見つかります:
    1. *OTB* タブを展開します
    2. *OTB* **フォルダ** を設定します。これは OTB がインストールされている場所です。
    3. *OTB* **アプリケーションフォルダ** を設定します。これは OTB アプリケーションの場所です（`<PATH_TO_OTB_INSTALLATION>/lib/otb/applications`）
    4. "ok"をクリックして設定を保存しダイアログを閉じます。

設定が正しければ **プロセシングツールボックス** で OTB アルゴリズムが利用可能になります。

QGIS プロセシングで利用できる OTB 設定のドキュメント

- **OTB フォルダ**: これは OTB が利用可能なディレクトリです。
- **OTB アプリケーションフォルダ**：これは OTB アプリケーションの場所です。

  複数のパスが許されます。
- **ログのレベル**（オプション）：OTB アプリケーションが使うロガーのレベル。

  ログのレベルはアルゴリズム実行中に表示される詳細の量を制御します。指定できるログのレベルの値は `INFO`、`WARNING`、`CRITICAL`、`DEBUG` です。この値はデフォルトでは `INFO` です。これは上級者向けの設定です。
- **使用する RAM の最大値**（オプション）：OTB アプリケーションはデフォルトで利用可能なすべてのシステム RAM を使います。

  しかし、このオプションを使って、特定の量の RAM（MB 単位）を使うように OTB に指示することができます。256 の値は OTB プロセシングプロバイダによって無視されます。これは上級者向けの設定です。
- **ジオイドファイル**（オプション）：ジオイドファイルのパス。

  このオプションは、OTB アプリケーションのパラメータ elev.dem.geoid と elev.geoid の値を設定します。この値をグローバルに設定することで、複数の処理アルゴリズムで値を共有することができます。デフォルトでは空。
- **SRTM タイルフォルダ**（オプション）：SRTM タイルが利用できるディレクトリ。

  SRTM データをローカルに保存し、処理中にファイルをダウンロードすることを避けることができます。このオプションは、OTB アプリケーションの elev.dem.path と elev.dem パラメータの値を設定します。この値をグローバルに設定することで、複数の処理アルゴリズムで値を共有することができます。デフォルトでは空。

QGIS と OTB バージョンの互換性

（OTB 6.6.1 以降の）すべての OTB バージョンは最新の QGIS バージョンと互換があります。

トラブルシューティング

QGIS プロセシングの OTB アプリケーションに問題がある場合は、OTB bug tracker に `qgis` ラベルを付けて issue を登録してください。

OTB と QGIS に関する追加情報は、**`こちら<https://www.orfeo-toolbox.org/CookBook-develop/QGISInterface.html>`_** にあります。

第27章　プロセシングプロバイダとアルゴリズム

ここでは、プロセシングアルゴリズムとそのパラメータ（ユーザインタフェースの章で説明したもの）について説明しています。

27.1 QGIS アルゴリズムプロバイダー

QGIS のアルゴリズムプロバイダーは、ほとんど QGIS の API だけを使用して、さまざまな解析とジオプロセシング操作を実装しています。このため、このプロバイダーのほとんどすべてのアルゴリズムは、追加設定なしですぐに使えるようになっています。

このプロバイダーには、プラグインからいくつかのアルゴリズムが取り入れられており、さらに独自のアルゴリズムも追加されています。

27.1.1 地図製作

点を地物に揃える

ポイント地物を、別の参照レイヤにある最も近い地物の向きに整列させるために必要な回転量を計算します。出力レイヤには新しいフィールドが追加され、最も近い参照地物に対する角度（時計回り、度単位）が入力されます。

オプションとして、出力レイヤのシンボロジについて、計算された回転フィールドをマーカーシンボルの回転に使用するよう自動的に設定することができます。必要に応じて、孤立したポイントが遠く離れた地物に対して整列しないよう、ポイントを整列させる際に使用する最大距離を設定することができます。

ヒント: このアルゴリズムは、建物のポイントシンボルを最も近い道路の方向に沿わせるように揃えるといった使用事例を想定しています。

☑ ライン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	回転を計算するポイント地物
参照レイヤ	REFERENCE_LAYER	[ベクタ：任意]	回転量を計算するための地物を検索するレイヤ
最大参照距離 オプション	MAX_DISTANCE	[数値] デフォルト：未設定	この距離以内に参照地物が見つからない場合には、ポイント地物に回転角は設定されません。
角度の属性名	FIELD_NAME	[文字列] デフォルト：'rotation'	回転量の値を格納するフィールド名です。
シンボルを自動適用	APPLY_SYMBOLOGY	[ブール値] デフォルト：True	地物のシンボルマーカーを角度フィールドの値で回転させます。
Aligned layer 出力	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	回転されたベクタレイヤの出力先を指定します。次のうちどれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
Aligned layer 出力	OUTPUT	[ベクタ：ポイント]	角度フィールド付きのポイントレイヤが追加されます。QGIS に読み込まれると、入力レイヤのシンボロジがデフォルトで適用され、マーカーシンボルにはデータ定義の回転が適用されます。

Python コード

Algorithm ID: `native:angletonearest`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

スタイルデータベースを統合

複数の QGIS スタイルデータベースを 1 つのスタイルデータベースに統合します。複数のデータベースに同じ種類で同じ名前のアイテムがある場合には、統合結果の出力データベースではこれらは一意な名前となるように名前が変更されます。

参考:

スタイルデータベースを作成

パラメータ

ラベル	名前	データ型	説明
入力データベース	`INPUT`	[ファイル] [リスト]	QGIS スタイルアイテムを含むファイル
統合するオブジェクト	`OBJECTS`	[列挙型] [リスト]	新しいスタイルデータベースに入れたい入力データベース内のスタイルアイテムの種類。以下の種類があります： • 0 --- **シンボル** • 1 --- **カラーランプ** • 2 --- **テキストフォーマット** • 3 --- **ラベル設定**
出力のスタイルデータベース	`OUTPUT`	[ファイル] デフォルト：[一時レイヤを作成]	選択したスタイルアイテムを統合した出力 `.XML` ファイル。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
カラーランプの数	COLORRAMPS	[数値]	
ラベル設定の数	LABELSETTINGS	[数値]	
出力のスタイルデータベース	OUTPUT	[ファイル]	選択したスタイルアイテムを統合した出力 .XML ファイル
シンボルの数	SYMBOLS	[数値]	
テキストフォーマットの数	TEXTFORMATS	[数値]	

Python コード

Algorithm ID: native:combinestyles

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

スタイルからカテゴリ値レンダラーを作成する

ベクタレイヤのレンダラを、スタイルデータベースのマッチするシンボルを使用したカテゴリ値レンダラに変更します。スタイルファイルを指定しない場合には、ユーザーの現在の **シンボルライブラリ** を代わりに使用します。

指定された式またはフィールドを使用してレンダラのカテゴリを作成します。各カテゴリは、指定された QGIS XML スタイル データベース内に存在するシンボルと個別にマッチングします。一致するシンボル名が見つかると、カテゴリのシンボルはこの一致したシンボルに設定されます。

必要ならば、シンボルとマッチしなかったカテゴリ名のリストや、カテゴリとマッチしなかったシンボル名のリストをテーブルに出力できます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	分類されたスタイルを適用するベクタレイヤ
式を使用したカテゴリ（分類）	FIELD	[式]	地物を分類するフィールドまたは式
スタイルデータベース（既存シンボルを使う場合は空白にしてください）	STYLE	[ファイル]	入力レイヤのカテゴリに適用するシンボルを含むファイル（.XML ファイル）。このファイルは、スタイルマネージャの **シンボルの共有** ツールを使用して取得することもできます。ファイルを指定しない場合には、QGIS のローカルなシンボルライブラリを使用します。
大文字小文字を区別する	CASE_SENSITIVE	[ブール値] デフォルト：False	True（チェック）の場合、カテゴリとシンボル名の間で大文字小文字を区別した比較を行います
非アルファベット文字は無視する	TOLERANT	[ブール値] デフォルト：False	True（チェック）の場合、カテゴリとシンボル名の非アルファベット文字は無視します。マッチング時の許容度が大きくなります。
一致しなかったカテゴリ オプション	NON_MATCHING_CAT	[テーブル] デフォルト：[出力をスキップ]	スタイルデータベース内のどのシンボルにもマッチしなかったカテゴリの出力テーブル。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
一致しなかったシンボル名 オプション	NON_MATCHING_SYM	[テーブル] デフォルト：[出力をスキップ]	どのカテゴリともマッチしなかったスタイルデータベースのシンボルの出力テーブル。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
一致しなかったカテゴリ	NON_MATCHING_CAT	[テーブル]	指定されたスタイルデータベースのどのシンボルともマッチしなかったカテゴリをリストします。
一致しなかったシンボル名	NON_MATCHING_SYM	[テーブル]	どのカテゴリともマッチしなかったスタイルデータベースのシンボルをリストします。
分類されたレイヤ	OUTPUT	[入力レイヤと同じ]	入力ベクタレイヤに分類されたスタイルが適用されたもの。新しいレイヤは出力されません。

Python コード

Algorithm ID: `native:categorizeusingstyle`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

スタイルデータベースを作成

QGIS のプロジェクトから全てのスタイルオブジェクト（シンボル、カラーランプ、テキストフォーマット、ラベル設定）を抽出します。

抽出されたシンボルは、**スタイルマネージャ** ダイアログで管理とインポートができる、QGIS スタイルデータベース（ XML 形式）に保存されます。

参考:

スタイルデータベースを統合

パラメータ

ラベル	名前	データ型	説明
入力プロジェクト（空白なら現在のプロジェクト） オプション	INPUT	[ファイル]	スタイルアイテムを抽出したい QGIS プロジェクトファイル
抽出するオブジェクト	OBJECTS	[列挙型] [リスト]	新しいスタイルデータベースに入れたい入力プロジェクト内のスタイルアイテムの種類。以下の種類があります： • 0 --- **シンボル** • 1 --- **カラーランプ** • 2 --- **テキストフォーマット** • 3 --- **ラベル設定**
出力のスタイルデータベース	OUTPUT	[ファイル] デフォルト：[一時レイヤを作成]	選択したスタイルアイテムの出力 .XML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
カラーランプの数	COLORRAMPS	[数値]	カラーランプの数
ラベル設定の数	LABELSETTINGS	[数値]	ラベル設定の数
出力のスタイルデータベース	OUTPUT	[ファイル]	選択したスタイルアイテムの出力 .XML ファイル
シンボルの数	SYMBOLS	[数値]	シンボルの数
テキストフォーマットの数	TEXTFORMATS	[数値]	テキストフォーマットの数

Python コード

Algorithm ID: native:stylefromproject

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地図帳レイアウトを画像として出力

印刷レイアウトの地図帳を画像ファイル（PNG 画像や JPEG 画像など）としてエクスポートします。

カバレッジレイヤが設定されている場合、選択したレイアウトの地図帳設定はこのアルゴリズムの設定で上書きされます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
地図帳レイアウト	LAYOUT	[レイアウト]	エクスポートしたいレイアウト
カバレッジ・レイヤ オプション	COVERAGE_LAYER	[ベクタ：任意]	地図帳を生成するために使用するレイヤ
フィルタ式	FILTER_EXPRESSIO	[式]	地図帳地物のフィルタリングに使用する式
ソート式 オプション	SORTBY_EXPRESSIO	[式]	地図帳地物のソートに使用する式
逆順ソート オプション	SORTBY_REVERSE	[ブール値]	ソートを逆順にするかを決定します。ソート式を指定した場合に使用します。
出力ファイル名の式	FILENAME_EXPRESS	[式] デフォルト：'out-put_'||@atlas_feature	ファイル名を生成するために使用する式
出力フォルダ	FOLDER	[フォルダ]	生成した画像の保存先フォルダ

詳細パラメータ

ラベル	名前	データ型	説明
地図アイテムに割り当てられる地図レイヤ オプション	LAYERS	[列挙型] [レイヤ]	地図アイテムに表示するレイヤ（ロックされているものは状態変更不可）
画像フォーマット	EXTENSION	[列挙型] デフォルト： png	生成する出力結果のファイル形式。利用可能な形式のリストは、OS とインストールされているドライバによって異なります。
DPI オプション	DPI デフォルト：未設定	[数値]	出力ファイルの DPI。設定なしの場合には、印刷レイアウトの設定値が使用されます。
ワールドファイルを生成	GEOREFERENCE	[ブール値] デフォルト：True	ワールドファイルを生成するかどうかの指定
RDF メタデータのエクスポート	INCLUDE_METADATA	[ブール値] デフォルト：True	RDF メタデータ（タイトル、著者など）を生成するかどうかの指定
アンチエイリアスを有効化	ANTIALIAS	[ブール値] デフォルト：True	アンチエイリアスを有効にするかどうかの指定

出力

ラベル	名前	データ型	説明
出力ファイル	OUTPUT	[ファイル]	地図帳レイアウトによって生成された画像ファイル

Python コード

Algorithm ID: `native:atlaslayouttoimage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地図帳レイアウトを複数の PDF として出力

NEW in 3.24

印刷レイアウトの地図帳を複数の PDF ファイルにエクスポートします。

カバレッジレイヤが設定されている場合、選択したレイアウトの地図帳設定はこのアルゴリズムの設定で上書きされます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
地図帳レイアウト	LAYOUT	[レイアウト]	エクスポートしたいレイアウト
カバレッジ・レイヤ オプション	COVERAGE_LAYER	[ベクタ：任意]	地図帳を生成するために使用するレイヤ
フィルタ式	FILTER_EXPRESSIO	[式]	地図帳地物のフィルタリングに使用する式
ソート式 オプション	SORTBY_EXPRESSIO	[式]	地図帳地物のソートに使用する式
逆順ソート オプション	SORTBY_REVERSE	[ブール値]	ソートを逆順にするかを決定します。ソート式を指定した場合に使用します。
出力ファイル名 オプション	OUTPUT_FILENAME	[式]	PDF 出力ファイルの名前のパターン。
出力フォルダ	OUTPUT_FOLDER	[フォルダ]	出力 PDF ファイルの保存先フォルダー。

詳細パラメータ

ラベル	名前	データ型	説明
地図アイテムに割り当てられる地図レイヤ オプション	LAYERS	[列挙型] [レイヤ]	地図アイテムに表示するレイヤ（ロックされているものは状態変更不可）
DPI オプション	DPI デフォルト：未設定	[数値]	出力ファイルの DPI。設定なしの場合には、印刷レイアウトの設定値が使用されます。
常にベクタとしてエクスポート	FORCE_VECTOR	[ブール値] デフォルト：False	ベクタデータを常にベクタのままとするかの指定
常にラスタとしてエクスポート NEW in 3.28	FORCE_RASTER	[ブール値] デフォルト：False	マップ内のすべてのアイテムを強制的にラスタライズします。このパラメータは FORCE_VECTOR パラメータよりも優先されます。
地理参照情報を追加	GEOREFERENCE	[ブール値] デフォルト：True	ワールドファイルを生成するかどうかの指定
RDF メタデータのエクスポート	INCLUDE_METADATA	[ブール値] デフォルト：True	RDF メタデータ（タイトル、著者など）を生成するかどうかの指定
ラスタタイルのエクスポートを無効化	DISABLE_TILED	[ブール値] デフォルト：False	ラスタ画像をタイル化するかどうかの指定
ジオメトリを簡略化してファイルを縮小する	SIMPLIFY	[ブール値] デフォルト：True	ファイルサイズを減らすためにジオメトリを簡略化するかどうかの指定
テキスト出力	TEXT_FORMAT	[列挙型] デフォルト： 0	テキストをパスとしてエクスポートするか、テキストオブジェクトとしてエクスポートするかの指定。オプションは次のとおり： • 0 - テキストを常にパスとして出力（推奨） • 1 - テキストを常にテキストオブジェクトとして出力
画像圧縮 NEW in 3.28	IMAGE_COMPRESSIO	[列挙型] デフォルト： 0	画像の圧縮レベルとファイルが印刷出力や外部アプリケーションでのポストプロダクションにどの程度適しているかを決定します。可能なオプションは次のとおりです: • 0 - Lossy (JPEG) • 1 - Lossless

出力

ラベル	名前	データ型	説明
PDF ファイル	OUTPUT	[ファイル]	エクスポートした地図帳レイアウトに対応する PDF ファイル

Python コード

Algorithm ID: native:atlaslayouttomultiplepdf

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地図帳レイアウトを PDF として出力

印刷レイアウトの地図帳を単独の PDF ファイルにエクスポートします。

カバレッジレイヤが設定されている場合、選択したレイアウトの地図帳設定はこのアルゴリズムの設定で上書きされます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
地図帳レイアウト	LAYOUT	[レイアウト]	エクスポートしたいレイアウト
カバレッジ・レイヤ オプション	COVERAGE_LAYER	[ベクタ：任意]	地図帳を生成するために使用するレイヤ
フィルタ式	FILTER_EXPRESSIO	[式]	地図帳地物のフィルタリングに使用する式
ソート式 オプション	SORTBY_EXPRESSIO	[式]	地図帳地物のソートに使用する式
逆順ソート オプション	SORTBY_REVERSE	[ブール値]	ソートを逆順にするかを決定します。ソート式を指定した場合に使用します。
PDF ファイル	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	（パスを含む）出力ファイルの名前。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
地図アイテムに割り当てられる地図レイヤ オプション	LAYERS	[列挙型] [レイヤ]	地図アイテムに表示するレイヤ（ロックされているものは状態変更不可）
DPI オプション	DPI デフォルト：未設定	[数値]	出力ファイルの DPI。設定なしの場合には、印刷レイアウトの設定値が使用されます。
常にベクタとしてエクスポート	FORCE_VECTOR	[ブール値] デフォルト：False	ベクタデータを常にベクタのままとするかの指定
常にラスタとしてエクスポート NEW in 3.28	FORCE_RASTER	[ブール値] デフォルト：False	マップ内のすべてのアイテムを強制的にラスタライズします。このパラメータは FORCE_VECTOR パラメータよりも優先されます。
地理参照情報を追加	GEOREFERENCE	[ブール値] デフォルト：True	ワールドファイルを生成するかどうかの指定
RDF メタデータのエクスポート	INCLUDE_METADATA	[ブール値] デフォルト：True	RDF メタデータ（タイトル、著者など）を生成するかどうかの指定
ラスタタイルのエクスポートを無効化	DISABLE_TILED	[ブール値] デフォルト：False	ラスタ画像をタイル化するかどうかの指定
ジオメトリを簡略化してファイルを縮小する	SIMPLIFY	[ブール値] デフォルト：True	ファイルサイズを減らすためにジオメトリを簡略化するかどうかの指定
テキスト出力	TEXT_FORMAT	[列挙型] デフォルト： 0	テキストをパスとしてエクスポートするか、テキストオブジェクトとしてエクスポートするかの指定。オプションは次のとおり： • 0 - テキストを常にパスとして出力（推奨） • 1 - テキストを常にテキストオブジェクトとして出力
画像圧縮 NEW in 3.28	IMAGE_COMPRESSIO	[列挙型] デフォルト： 0	画像の圧縮レベルとファイルが印刷出力や外部アプリケーションでのポストプロダクションにどの程度適しているかを決定します。可能なオプションは次のとおりです: • 0 - Lossy (JPEG) • 1 - Lossless

出力

ラベル	名前	データ型	説明
PDF ファイル	OUTPUT	[ファイル]	エクスポートした地図帳レイアウトに対応する PDF ファイル

Python コード

Algorithm ID: native:atlaslayouttopdf

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイアウトを画像として出力

印刷レイアウトを画像ファイル（PNG 画像や JPEG 画像など）としてエクスポートします。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
印刷レイアウト	LAYOUT	[レイアウト]	エクスポートしたいレイアウト
出力ファイル	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	（パスを含む）出力ファイルの名前。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
地図アイテムに割り当てられる地図レイヤ オプション	LAYERS	[列挙型] [レイヤ]	地図アイテムに表示するレイヤ（ロックされているものは状態変更不可）
DPI オプション	DPI デフォルト：未設定	[数値]	出力ファイルの DPI。設定なしの場合には、印刷レイアウトの設定値が使用されます。
ワールドファイルを生成	GEOREFERENCE	[ブール値] デフォルト：True	ワールドファイルを生成するかどうかの指定
RDF メタデータのエクスポート	INCLUDE_METADATA	[ブール値] デフォルト：True	RDF メタデータ（タイトル、著者など）を生成するかどうかの指定
アンチエイリアスを有効化	ANTIALIAS	[ブール値] デフォルト：True	アンチエイリアスを有効にするかどうかの指定

出力

ラベル	名前	データ型	説明
出力ファイル	OUTPUT	[ファイル]	エクスポートした印刷レイアウトに対応する画像ファイル

Python コード

Algorithm ID: `native:printlayouttoimage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイアウトを PDF として出力

印刷レイアウトを PDF ファイルとしてエクスポートします。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
印刷レイアウト	LAYOUT	[レイアウト]	エクスポートしたいレイアウト
PDF ファイル	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	（パスを含む）出力ファイルの名前。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
地図アイテムに割り当てられる地図レイヤ オプション	LAYERS	[列挙型] [レイヤ]	地図アイテムに表示するレイヤ（ロックされているものは状態変更不可）
DPI オプション	DPI デフォルト：未設定	[数値]	出力ファイルの DPI。設定なしの場合には、印刷レイアウトの設定値が使用されます。
常にベクタとしてエクスポート	FORCE_VECTOR	[ブール値] デフォルト：False	ベクタデータを常にベクタのままとするかの指定
常にラスタとしてエクスポート NEW in 3.28	FORCE_RASTER	[ブール値] デフォルト：False	マップ内のすべてのアイテムを強制的にラスタライズします。このパラメータは FORCE_VECTOR パラメータよりも優先されます。
地理参照情報を追加	GEOREFERENCE	[ブール値] デフォルト：True	ワールドファイルを生成するかどうかの指定
RDF メタデータのエクスポート	INCLUDE_METADATA	[ブール値] デフォルト：True	RDF メタデータ（タイトル、著者など）を生成するかどうかの指定
ラスタタイルのエクスポートを無効化	DISABLE_TILED	[ブール値] デフォルト：False	ラスタ画像をタイル化するかどうかの指定
ジオメトリを簡略化してファイルを縮小する	SIMPLIFY	[ブール値] デフォルト：True	ファイルサイズを減らすためにジオメトリを簡略化するかどうかの指定
テキスト出力	TEXT_FORMAT	[列挙型] デフォルト： 0	テキストをパスとしてエクスポートするか、テキストオブジェクトとしてエクスポートするかの指定。オプションは次のとおり： • 0 - テキストを常にパスとして出力（推奨） • 1 - テキストを常にテキストオブジェクトとして出力
画像圧縮 NEW in 3.28	IMAGE_COMPRESSIO	[列挙型] デフォルト： 0	画像の圧縮レベルとファイルが印刷出力や外部アプリケーションでのポストプロダクションにどの程度適しているかを決定します。可能なオプションは次のとおりです: • 0 - Lossy (JPEG) • 1 - Lossless
レイヤを別々の PDF ファイルに出力	SEPARATE_LAYERS	[ブール値] デフォルト：False	True の場合には、レイアウト内の地図アイテムごと、レイヤごとに、別々の PDF ファイルが作成されます。さらに、他の複雑なレイアウトアイテムに対して個別の PDF ファイルを作成するため、レイアウトの論理的な原子コンポーネントを含む PDF ファイルのセットが作成され

出力

ラベル	名前	データ型	説明
PDF ファイル	OUTPUT	[ファイル]	エクスポートした印刷レイアウトに対応する PDF ファイル

Python コード

Algorithm ID: native:printlayouttopdf

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラベルを抽出

NEW in 3.24

指定された範囲と縮尺でレンダリングされた地図からラベルの情報を抽出します。

地図テーマが与えられた場合、レンダリングされた地図はそのテーマの可視性とシンボロジに一致します。空白の場合、プロジェクトのすべての可視レイヤが使用されます。抽出されたラベル情報には、位置（点ジオメトリとして提供される）、関連するレイヤ名と地物 ID、ラベルテキスト、回転（度単位、時計回り）、複数行の整列、およびフォントの詳細が含まれます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
地図の領域	EXTENT	[範囲]	ラベルを抽出する地図の範囲 利用できる方法: • レイヤから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします • `xmin, xmax, ymin, ymax` として座標を入力
地図の縮尺	SCALE	[scale]	抽出されたラベルはこの縮尺で設定されたプロパティを使ってレンダリングされます
地図のテーマ オプション	MAP_THEME	[maptheme]	ラベルを抽出するレイヤを表示するマップテーマ。未設定の場合、現在表示されているレイヤのラベルが抽出されます。
未配置のラベルを含む オプション	INCLUDE_UNPLACED	[ブール値] デフォルト：True	競合する（そのため配置されていない）ラベルも含めて、重複するラベルをすべて抽出するかどうかを指定します。
抽出ラベル	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	領域の出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
地図の解像度 (DPI) オプション	`DPI` デフォルト: 96.0	[数値]	

出力

ラベル	名前	データ型	説明
抽出ラベル	OUTPUT	[ベクタ：ポイント]	取得したラベルを表す点ベクタレイヤ。各地物は、そのソース（レイヤ、地物 ID）と割り当てられたラベリング プロパティ（テキスト、フォント、サイズ、回転など）を識別する属性を持ちます。ラベリングと null シンボルのデフォルトスタイルもレイヤに適用されます。

警告: 生成されたフィールドの中には 10 文字以上の名前を持つものがあるため、ESRI のシェープフ

Python コード

Algorithm ID: `native:extractlabels`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

印刷レイアウト領域を作成

印刷レイアウトの地図アイテム（複数可）の表示範囲を示すポリゴンを作成します。ポリゴンは、地図のサイズ（レイアウト単位、すなわち **参照マップ** の単位）や縮尺、回転を属性として持ちます。

「地図のアイテム」をパラメータで指定した場合には、その地図の範囲だけが出力されます。指定しない場合には、レイアウト上のすべての地図が出力されます。

オプションとして、出力 CDS を指定できます。指定しない場合、元の地図アイテムの CRS が使用されます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
印刷レイアウト	LAYOUT	[列挙型]	現在のプロジェクトの印刷レイアウト
地図のアイテム オプション	MAP	[列挙型] デフォルト： **すべての地図アイテム**	情報を抽出したい地図アイテム（複数可）。パラメータを指定しない場合には、すべての地図アイテムが処理されます。
領域	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	領域の出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
CRS の上書き オプション	CRS	[crs] デフォルト： **レイアウトの** *CRS*	情報が報告されるレイヤの CRS を選択します。

出力

ラベル	名前	データ型	説明
地図の高さ	HEIGHT	[数値]	
領域	OUTPUT	[ベクタ：ポリゴン]	入力印刷レイアウトのすべての地図アイテムの範囲を含むポリゴンベクタレイヤの出力結果
地図の回転	ROTATION	[数値]	
地図の縮尺	SCALE	[数値]	
地図の幅	WIDTH	[数値]	

Python コード

Algorithm ID: `native:printlayoutmapextenttolayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤスタイルの設定

レイヤに指定されたスタイルを適用します。スタイルは QML ファイルで定義します。

新しい出力結果は作成されません。スタイルはレイヤに即座に適用されます。

パラメータ

ラベル	名前	データ型	説明
レイヤ	INPUT	[レイヤ]	スタイルを適用させたい入力レイヤ
スタイルファイル（QML ファイル）	STYLE	[ファイル]	スタイルの .qml ファイルへのパス

出力

ラベル	名前	データ型	説明
	OUTPUT	[入力レイヤと同じ]	入力レイヤに新しいスタイルが適用されたもの。新しいレイヤは作成されません。

Python コード

Algorithm ID: native:setlayerstyle

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

トポロジーによる塗り分け

隣接するポリゴンが同じ色インデックスとならないように、ただし、必要な色インデックスの数は最小となるように、ポリゴン地物に色インデックスを割り当てます。

このアルゴリズムでは、色を割り当てる際の方法を選択できます。

必要に応じて、最小色数を指定できます。色インデックスは **color_id** という名前の新しい属性に保存されます。

以下の例では、色 4 つを選択した場合のアルゴリズムを示しています。各色クラスの地物の数が同じになるように割り当てられていることがわかります。

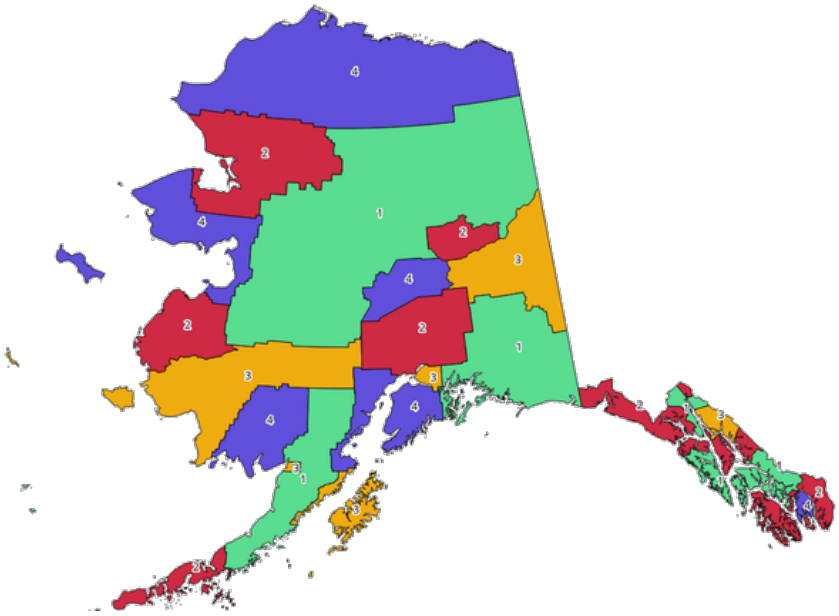


図 27.1: トポロジーによる塗り分けの例

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ポリゴンレイヤ
最小色数	MIN_COLORS	[数値] デフォルト： 4	割り当てる最小の色数。最小値は 1、最大値は 1000
同じ色の地物の間の最小距離	MIN_DISTANCE	[数値] デフォルト： 0.0	近くにある（しかし接触していない）地物に同じ色が割り当てられないようにします。最小値は 0.0 です。

次のページに続く

表 27.2 – 前のページからの続き

ラベル	名前	データ型	説明
色を割り当てる方針	BALANCE	[列挙型] デフォルト： 0	選択肢は以下のとおりです： • 0 --- 地物の数が同じになるように割り当て 各色インデックスに割り当てられた地物の数が均衡するように色を割り当てます。 • 1 --- 面積が同じになるように割り当て 各色に割り当てられた地物の面積の合計が均衡するように色を割り当てます。このモードは便利です。このモードは、巨大な地物があると、塗り分けられた地図上である色が支配的になっているように見えることを避けるのに便利です。 • 2 --- 同じ色の地物ができるだけ離れるように割り当て 同じ色の地物間の距離が最大になるように色を割り当てます。このモードでは、地図上の色の分布をより均一にできます。
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	color_id カラムが追加されたポリゴンベクタレイヤ

Python コード

Algorithm ID: `qgis:topologicalcoloring`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メインレイヤの注記を転送

プロジェクトのメイン注記レイヤから新しい注記レイヤに、すべての :ref:` 注記 <annotation_layer>` を転送します。アイテムの配置は、レイヤスタック内で調整することができます。

パラメータ

ラベル	名前	データ型	説明
新規レイヤの名前	LAYER_NAME	[文字列] デフォルト: '注記'	作成する注記レイヤの名前

出力

ラベル	名前	データ型	説明
新規レイヤの名前	OUTPUT	[レイヤ]	メインの注記レイヤからのアイテムを持つレイヤ

Python コード

Algorithm ID: `native:transferannotationsfrommain`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.2 データベース

PostgreSQL へエクスポート

ベクタレイヤを PostgreSQL データベースにインポートし、テーブルを作成します。同じ名前のテーブルがある場合には、新たなテーブルを作成する前に、このテーブルは削除されます。これを使用する前に、QGIS と PostgreSQL データベース間の接続を作成（ **保存された接続の作成** 参照）しておく必要があります。

パラメータ

ラベル	名前	データ型	説明
エクスポートするレイヤ	INPUT	[ベクタ：任意]	データベースに追加したいベクタレイヤ
データベース（接続名）	DATABASE	[文字列]	データベース接続の名前（データベース名ではありません）。既存の接続がコンボボックスに表示されます。
スキーマ（スキーマ名） オプション	SCHEMA	[文字列] デフォルト：'public'	データを保存するスキーマの名前。新しいスキーマ、既存のスキーマのいずれも可能です。
インポートするテーブル（空白の場合、レイヤ名が使われます） オプション	TABLENAME	[文字列] デフォルト：''	インポートするベクタファイルのテーブル名を定義します。何も指定しない場合は、レイヤ名が使用されます。
主キーの属性（フィールド） オプション	PRIMARY_KEY	[テーブルのフィールド：任意]	ベクタレイヤの既存のフィールドから主キーフィールドを設定します。**一意な** 値を持つカラムを主キーとして使うことができます。
ジオメトリのカラム	GEOMETRY_COLUMN	[文字列] デフォルト：'geom'	新しい PostGIS のテーブルのジオメトリカラムの名前を定義します。地物のジオメトリ情報は、このカラムに格納されます。
文字コード オプション	ENCODING	[文字列] デフォルト：'UTF-8'	出力レイヤの文字コードを定義します
上書き	OVERWRITE	[ブール値] デフォルト： True	指定されたテーブルが存在する場合、このオプションを True とすると、そのテーブルは削除され、地物を追加する前に新しいテーブルが作成されます。このオプションが False でテーブルが既に存在する場合、このアルゴリズムは例外（"relation already exists"）を投げます。
空間インデックスを作成	CREATEINDEX	[ブール値] デフォルト： True	空間インデックスを作成するかどうかを指定します

次のページに続く

表 27.4 – 前のページからの続き

ラベル	名前	データ型	説明
フィールド名を小文字に変換	LOWERCASE_NAMES	[ブール値] デフォルト： True	入力ベクタレイヤのフィールド名を小文字に変換します
文字フィールドの長さ制約を解く	DROP_STRING_LENG	[ブール値] デフォルト: False	文字フィールドの長さ制約を解くか否か
マルチパートの代わりにシングルパートのジオメトリを作る	FORCE_SINGLEPART	[ブール値] デフォルト: False	出力レイヤの地物をマルチパートではなくシングルパートとするかどうか。デフォルトでは、既存のジオメトリ情報が保持されます。

出力

このアルゴリズムに出力はありません。

Python コード

Algorithm ID: `qgis:importintopostgis`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

SpatiaLite へ出力

ベクタレイヤを SpatiaLite データベースにエクスポートします。これを使用する前に、QGIS と SpatiaLite データベース間の接続を作成（ *SpatiaLite* **レイヤ** 参照）しておく必要があります。

パラメータ

ラベル	名前	データ型	説明
エクスポートするレイヤ	INPUT	[ベクタ：任意]	データベースに追加したいベクタレイヤ
データベースのファイル	DATABASE	[ベクタ：任意]	接続する SQLite または SpatiaLite データベースファイル

次のページに続く

表 27.5 – 前のページからの続き

ラベル	名前	データ型	説明
インポートするテーブル（空白の場合、レイヤ名が使われます） オプション	TABLENAME	[文字列] デフォルト： ''	インポートされるベクタファイルのテーブル名を定義します。何も指定しない場合は、レイヤ名が使用されます。
主キーの属性（フィールド） オプション	PRIMARY_KEY	[テーブルのフィールド：任意]	入力ベクタレイヤのフィールドを主キーとして使用します
ジオメトリのカラム	GEOMETRY_COLUMN	[文字列] デフォルト：'geom'	新しい SpatiaLite のテーブルのジオメトリカラムの名前を定義します。地物のジオメトリ情報は、このカラムに格納されます。
文字コード オプション	ENCODING	[文字列] デフォルト：'UTF-8'	出力レイヤの文字コードを定義します
上書き	OVERWRITE	[ブール値] デフォルト： True	指定されたテーブルが存在する場合、このオプションを True とすると、そのテーブルは削除され、レイヤの地物が追加される前に新しいテーブルが作成されます。このオプションが False でテーブルが既に存在する場合、このアルゴリズムは例外（"table already exists"）を投げます。
空間インデックスを作成	CREATEINDEX	[ブール値] デフォルト： True	空間インデックスを作成するかどうかを指定します
フィールド名を小文字に変換	LOWERCASE_NAMES	[ブール値] デフォルト： True	入力ベクタレイヤのフィールド名を小文字に変換します。
文字フィールドの長さ制約を解く	DROP_STRING_LENG	[ブール値] デフォルト: False	文字フィールドの長さ制約を解くか否か
マルチパートの代わりにシングルパートのジオメトリを作る	FORCE_SINGLEPART	[ブール値] デフォルト: False	出力レイヤの地物をマルチパートではなくシングルパートとするかどうか。デフォルトでは、既存のジオメトリ情報が保持されます。

出力

このアルゴリズムに出力はありません。

Python コード

Algorithm ID: `qgis:importintospatialite`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤを GeoPackage 化

レイヤを GeoPackage に追加します。

GeoPackage が既に存在し、 既存の `GeoPackage` を上書きする にチェックを入れている場合には、これは上書き（削除してから再度作成）されます。GeoPackage が既に存在し、 既存の `GeoPackage` を上書きする にチェックを入れていない場合には、レイヤは追加されます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	LAYERS	[ベクタ：任意] [リスト]	GeoPackage にインポートする（ベクタ）レイヤ。ラスタレイヤはサポートしていません。ラスタレイヤを追加した場合には、 QgsProcessingException 例外が発生します。
既存の GeoPackage を上書きする	OVERWRITE	[ブール値] デフォルト: False	指定された GeoPackage が存在する場合、このオプションを True に設定すると、それが削除されて新しいものが作成されてからレイヤが加えられます。False に設定すると、レイヤは追加されます。
レイヤスタイルを GeoPackage に保存	SAVE_STYLES	[ブール値] デフォルト： True	レイヤスタイルを保存します
選択地物のみ保存	SELECTED_FEATURE	[ブール値] デフォルト: False	レイヤに選択範囲がある場合、このオプションに True を設定すると、選択されている地物のみが保存されます。選択範囲のないレイヤは、全ての地物が保存されます。
プロジェクトで定義されているリレーションのあるレイヤをエクスポート NEW in 3.28	EXPORT_RELATED_L	[ブール値] デフォルト: False	入力レイヤにプロジェクトで設定された *relations* がある場合、このオプションを True に設定すると、関連したレイヤもエクスポートされます。レイヤで地物が選択されている場合、関連したレイヤも入力レイヤでない限り、関連した地物のみがエクスポートされます。
GeoPackage のファイル名	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	GeoPackage ファイルの保存場所を指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
新しい GeoPackage のレイヤ	OUTPUT_LAYERS	[文字列] [リスト]	GeoPackage に追加されたレイヤのリスト

Python コード

Algorithm ID: native:package

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

PostGIS を実行し読み込む

QGIS に接続された PostgreSQL データベースで SQL データベースクエリを実行し、その結果を読み込みます。このアルゴリズムは、新しいレイヤを作成 **しません**。レイヤ自身に対してクエリを実行するように設計されています。

例

1. ある既存のフィールドのすべての値を固定値に設定します。SQL クエリ文字列は次のようになります：

   ```
   UPDATE your_table SET field_to_update=20;
   ```

   上の例では、テーブル your_table のフィールド field_to_update の値が全て 20 に設定されます。

2. 新しく area カラムを作成し、ST_AREA PostGIS 関数を使用して各地物の面積を計算します。

   ```
   -- Create the new column "area" on the table your_table"
   ALTER TABLE your_table ADD COLUMN area double precision;
   -- Update the "area" column and calculate the area of each feature:
   UPDATE your_table SET area=ST_AREA(geom);
   ```

参考:

PostgreSQL で *SQL* を実行, *SQL* の実行, *SpatiaLite* を実行

パラメータ

ラベル	名前	データ型	説明
データベース（接続名）	DATABASE	[文字列]	データベース接続（データベース名ではありません）。既存の接続がコンボボックスに表示されます。
SQL クエリ	SQL	[文字列]	SQL クエリを定義します。例えば、`'UPDATE my_table SET field=10'`。
一意の ID のフィールド名	ID_FIELD	[文字列] デフォルト： id	主キーのフィールド（結果テーブルのカラム）を設定します
ジオメトリのフィールド名 オプション	GEOMETRY_FIELD	[文字列] デフォルト：'geom'	ジオメトリカラムの名前（結果テーブルのカラム）

出力

ラベル	名前	データ型	説明
SQL レイヤ	OUTPUT	[ベクタ：任意]	QGIS に読み込まれる結果のベクタレイヤ

Python コード

Algorithm ID: `qgis:postgisexecuteandloadsql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

PostgreSQL で SQL を実行

QGIS に接続された PostgreSQL データベースで SQL データベースクエリを実行します。このアルゴリズムは、新しいレイヤを作成 **しません**。レイヤ自身に対してクエリを実行するように設計されています。

例

1. ある既存のフィールドのすべての値を固定値に設定します。SQL クエリ文字列は次のようになります：

```
UPDATE your_table SET field_to_update=20;
```

上の例では、テーブル your_table のフィールド field_to_update の値が全て 20 に設定されます。

2. 新しく area カラムを作成し、ST_AREA PostGIS 関数を使用して各地物の面積を計算します。

```
-- Create the new column "area" on the table your_table"
ALTER TABLE your_table ADD COLUMN area double precision;
-- Update the "area" column and calculate the area of each feature:
UPDATE your_table SET area=ST_AREA(geom);
```

参考:

PostGIS **を実行し読み込む**, *SQL* **の実行**, *SpatiaLite* **を実行**

パラメータ

ラベル	名前	データ型	説明
データベース（接続名）	DATABASE	[文字列]	データベース接続（データベース名ではありません）。既存の接続がコンボボックスに表示されます。
SQL クエリ	SQL	[文字列]	SQL クエリを定義します。例えば、'UPDATE my_table SET field=10'。

出力

出力は作成されません。SQL クエリが実行されます。

Python コード

Algorithm ID: native:postgisexecutesql

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

SpatiaLite を実行

SpatiaLite データベースで SQL データベースクエリを実行します。このアルゴリズムは、新しいレイヤを作成 **しません**。レイヤ自身に対してクエリを実行するように設計されています。

参考:

PostgreSQL で *SQL* を実行, *SQL* の実行

SQL クエリの例については、:ref:`PostGIS SQL クエリの例 <qgis_postgis_execute_sql_example>`を参照してください。

パラメータ

ラベル	名前	データ型	説明
データベース	`DATABASE`	[ベクタ]	接続する SQLite または SpatiaLite データベースファイル
SQL クエリ	`SQL`	[文字列] デフォルト：''	SQL クエリを定義します。例えば、`'UPDATE my_table SET field=10'`。

出力

出力は作成されません。SQL クエリが実行されます。

Python コード

Algorithm ID: `native:spatialiteexecutesql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

SpatiaLite で実行（登録済み DB）

QGIS に接続された SpatiaLite データベースで SQL データベースクエリを実行します。このアルゴリズムは新しいレイヤを作成 **しません**。レイヤ自身に対してクエリを実行するように設計されています。

参考:

PostgreSQL で *SQL* を実行, *SQL* の実行

SQL クエリの例については、:ref:`PostGIS SQL クエリの例 <qgis_postgis_execute_sql_example>`を参照してください。

パラメータ

ラベル	名前	データ型	説明
データベース	`DATABASE`	[列挙型] デフォルト：未設定	現在のセッションに接続している SQLite/SpatiaLite データベースを選択します
SQL クエリ	`SQL`	[文字列] デフォルト：''	SQL クエリを定義します。例えば、`'UPDATE my_table SET field=10'`。

出力

出力は作成されません。SQL クエリが実行されます。

Python コード

Algorithm ID: `native:spatialiteexecutesqlregistered`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.3 ファイルツール

ファイルのダウンロード

URL（例えば `http:` や `file:` ）を使って指定されるファイルをダウンロードします。つまり、URL をコピー＆ペーストしてファイルをダウンロードできます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
URL	URL	[文字列]	ダウンロードしたいファイルの URL
ファイルのダウンロード先 オプション	OUTPUT	[文字列] デフォルト：[一時ファイルに保存]	ファイルの保存先の指定。次のいずれかです： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
方法	METHOD	[列挙型] デフォルト：0	リクエストに使う HTTP メソッド。選択肢は： • 0 --- GET • 1 --- POST
Data オプション	DATA	[文字列]	リクエストが POST の場合に本体に加えるデータ。

出力

ラベル	名前	データ型	説明
ファイルのダウンロード先	OUTPUT	[文字列]	ダウンロードしたファイルの保存場所

Python コード

Algorithm ID: `qgis:filedownloader`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.4 GPS

GPS データを変換

GPSBabel ツール を使って、GPS データファイルをさまざまなフォーマットから GPX 標準フォーマットに変換します。

パラメータ

ラベル	名前	データ型	説明
入力ファイル	INPUT	[ファイル]	変換するデータを格納しているファイル
形式	FORMAT	[列挙型]	変換するファイルの形式で、このリストにあるもの。
地物型	FEATURE_TYPE	[列挙型] デフォルト： 0	変換するデータの型 • 0 -- ウェイポイント • 1 -- 経路 • 2 -- 軌跡
出力	OUTPUT	[ベクタ：任意] デフォルト：[一時ファイルに保存]	GPX 出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
オーバーラップ	OUTPUT_LAYER	[ベクタ：任意]	GPX 標準形式のデータを持つ出力レイヤ

Python コード

Algorithm ID: native:convertgpsdata

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

GPX 地物型を変換

GPSBabel ツール を使って、GPX 地物をある型から別の型に変換します（例えば、すべてのウェイポイント地物をルートに変換します）。

パラメータ

ラベル	名前	データ型	説明
入力ファイル	INPUT	[ファイル]	変換するデータを格納しているファイル
変換	CONVERSION	[列挙型] デフォルト： 0	適用する変換の型 • 0 -- 経路からのウェイポイント • 1 -- 軌跡からのウェイポイント • 2 -- ウェイポイントからの経路 • 3 -- ウェイポイントからの軌跡
出力	OUTPUT	[vector: ポイントまたはライン] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力	OUTPUT	[ベクタ：任意]	変換された GPX 地物を出力するレイヤ

Python コード

Algorithm ID: native:convertgpxfeaturetype

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

GPS データをデバイスからダウンロード

GPSBabel ツール を使って、GPS デバイスから GPX 標準フォーマットにデータをダウンロードします。

パラメータ

ラベル	名前	データ型	説明
デバイス	DEVICE	[列挙型] デフォルト: *Garmin serial*	データを作成するために使った GPS デバイス。*GPS* **設定** ダイアログで宣言する必要があります。
ポート	PORT	[列挙型]	デバイスが接続されているポート。使用可能なポートは OS によって異なります。
地物型	FEATURE_TYPE	[列挙型] デフォルト：0	変換するデータの型 • 0 -- ウェイポイント • 1 -- 経路 • 2 -- 軌跡
出力	OUTPUT	[ベクタ：任意] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
オーバーラップ	OUTPUT_LAYER	[ベクタ：任意]	GPX 標準形式のデータを持つ出力レイヤ

Python コード

Algorithm ID: native:downloadgpsdata

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

GPS データをデバイスにアップロード

GPSBabel ツール を使って、GPX 標準フォーマットから GPS デバイスにデータをアップロードします。

パラメータ

ラベル	名前	データ型	説明
入力ファイル	INPUT	[ファイル]	アップロードするデータを格納している .GPX ファイル
デバイス	DEVICE	[列挙型] デフォルト: *Garmin serial*	データをアップロードしたい GPS デバイス。*GPS* **設定** ダイアログで宣言する必要があります。
ポート	PORT	[列挙型]	デバイスが接続されているポート。使用可能なポートは OS によって異なります。
地物型	FEATURE_TYPE	[列挙型] デフォルト： 0	アップロードするデータの型 • 0 -- ウェイポイント • 1 -- 経路 • 2 -- 軌跡

出力

出力はありません。成功するとデータはデバイスに読み込まれます。

Python コード

Algorithm ID: `native:uploadgpsdata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.5 内挿

ヒートマップ（カーネル密度推定）

カーネル密度推定を使用して、入力ポイントベクタレイヤの密度（ヒートマップ）ラスタを生成します。

密度はその場所にあるポイントの数に基づいて計算され、集まっているポイントの数が多ければ多いほど、値は大きくなります。ヒートマップを用いれば、 **ホットスポット** や点のクラスター化を簡単に確認できます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ（点）	INPUT	[ベクタ：ポイント]	ヒートマップに使用するポイントベクタレイヤ
半径	RADIUS	[数値] デフォルト：100.0	地図単位のヒートマップの検索半径（またはカーネルのバンド幅）。この半径は、ある点の周りで影響を受ける点の距離を指定します。大きな値を指定するほどヒートマップは滑らかになり、小さな値を指定すると、細部や点の密度のばらつきが表されます。

次のページに続く

表 27.13 – 前のページからの続き

ラベル	名前	データ型	説明
出力ラスタサイズ	PIXEL_SIZE	[数値] デフォルト： 0.1	レイヤの単位で表した、出力ラスタレイヤのピクセルサイズ GUI では、出力ラスタサイズは行数（行）/ 列数（カラム）で指定するか、**または** ピクセルサイズ（ピクセルサイズ X / ピクセルサイズ Y）で指定します。行数や列数が大きくなると、ピクセルサイズは小さくなり、出力ラスタのファイルサイズは増加します。 行 、 カラム 、 ピクセルサイズ X 、 ピクセルサイズ Y の値は同時に更新されます。つまり、行数を 2 倍にすれば列数も 2 倍になり、ピクセルサイズは半分になります。出力ラスタの範囲は（ほぼ）変わりません。
半径を示す属性（フィールド） オプション	RADIUS_FIELD	[テーブルのフィールド：数値]	入力レイヤの属性フィールドから、各地物の検索半径を設定します。
重みに使う属性 オプション	WEIGHT_FIELD	[テーブルのフィールド：数値]	入力地物を属性フィールドの値で重みづけします。結果のヒートマップにおいて、特定の地物の影響を増加させるために使用します。
カーネル関数	KERNEL	[列挙型] デフォルト： *0*	ポイントからの距離が長くなるにつれてポイントの影響力が減少する割合を制御します。カーネルによって減衰の速度は異なり、Triweight カーネルは Epanechnikov カーネルよりもポイントに近い距離の地物に大きな重みを与えます。このため、Triweight カーネルは「よりシャープな」、Epanechnikov カーネルは「よりスムーズな」ホットスポットになります。さまざまなカーネル関数が利用可能です（詳細については Wikipedia を参照してください）： • 0 --- Quartic • 1 --- Triangular • 2 --- Uniform • 3 --- Triweight • 4 --- Epanechnikov

次のページに続く

表 27.13 – 前のページからの続き

ラベル	名前	データ型	説明
減衰比（Triangular のみ） オプション	DECAY	[数値] デフォルト： *0.0*	Triangular カーネルで使用します。 ある地物からの熱が距離に従ってどのように減少するかをさらに制御できます。 • 値 0（=最小値）の場合は、与えられたの半径の中心に熱が集中し、端では完全に消滅すること意味します。 • 値 0.5 の場合は、半径の端のピクセルは、検索半径の中心のピクセルの半分の熱を持っていることを意味します。 • 値 1 の場合は、熱が検索半径の円全体に均等に分散していることを意味します（これは「Uniform」カーネルと同等です）。 • 値が 1 よりも大きい場合は、中心よりも検索半径の端の熱が高く、端に行くにつれて熱が高くなることを意味します。
スケーリングするか	OUTPUT_VALUE	[列挙型] デフォルト： **そのまま**	出力ヒートマップラスタの値を変更できます。次のいずれかです： • 0 --- そのまま • 1 --- スケールする
ヒートマップ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	カーネル密度値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ヒートマップ	OUTPUT	[ラスタ]	カーネル密度値のラスタレイヤ

例：ヒートマップの作成

以下の例では、QGIS サンプルデータセット（ **サンプルデータのダウンロード** 参照）の `airports` ベクタポイントレイヤを使用します。ヒートマップ作成に関する素晴らしい QGIS チュートリアルは、http://qgistutorials.com にもあります。

図 27.2 では、アラスカの空港が表示されています。

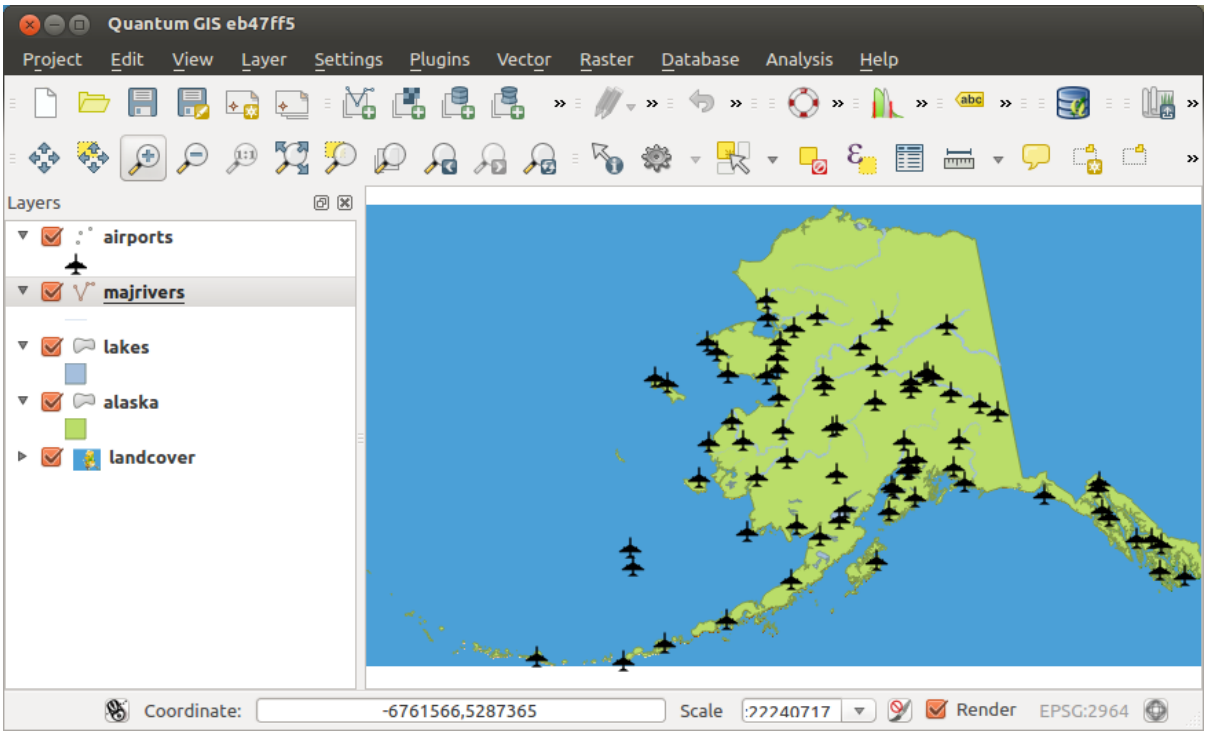


図 27.2: アラスカの空港

1. QGIS アルゴリズムの **内挿** グループから **ヒートマップ（カーネル密度推定）** アルゴリズム開きます
2. **入力レイヤ（点）** フィールドで、現在のプロジェクトにロードされたポイントレイヤのリストの中から `airports` レイヤを選択します。
3. **半径** を `1000000` メートルに変更します。
4. **ピクセルサイズ** *X* を `1000` に変更します。 **ピクセルサイズ** *Y* 、**行**、**カラム** は自動的に更新されます。
5. **実行** をクリックして、airports のヒートマップを作成し読み込みます（ 図 27.4 参照）。

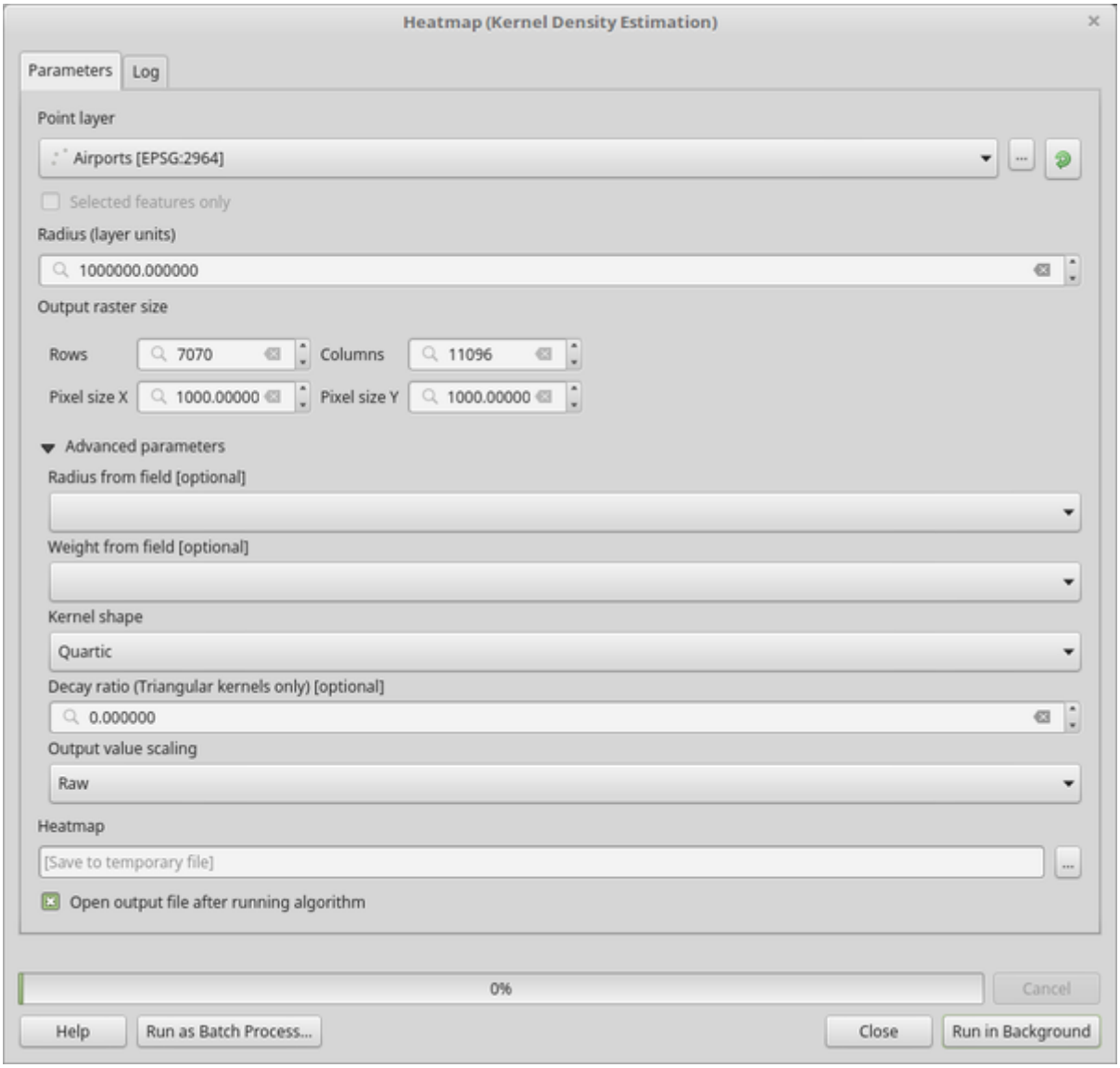


図 27.3: ヒートマップダイアログ

QGIS はヒートマップを生成し、マップウィンドウに追加します。デフォルトでは、ヒートマップはグレースケールで表示され、色が明るい領域は空港が集中していることを示しています。QGIS でこのヒートマップのスタイルを設定し、見た目を改善しましょう。

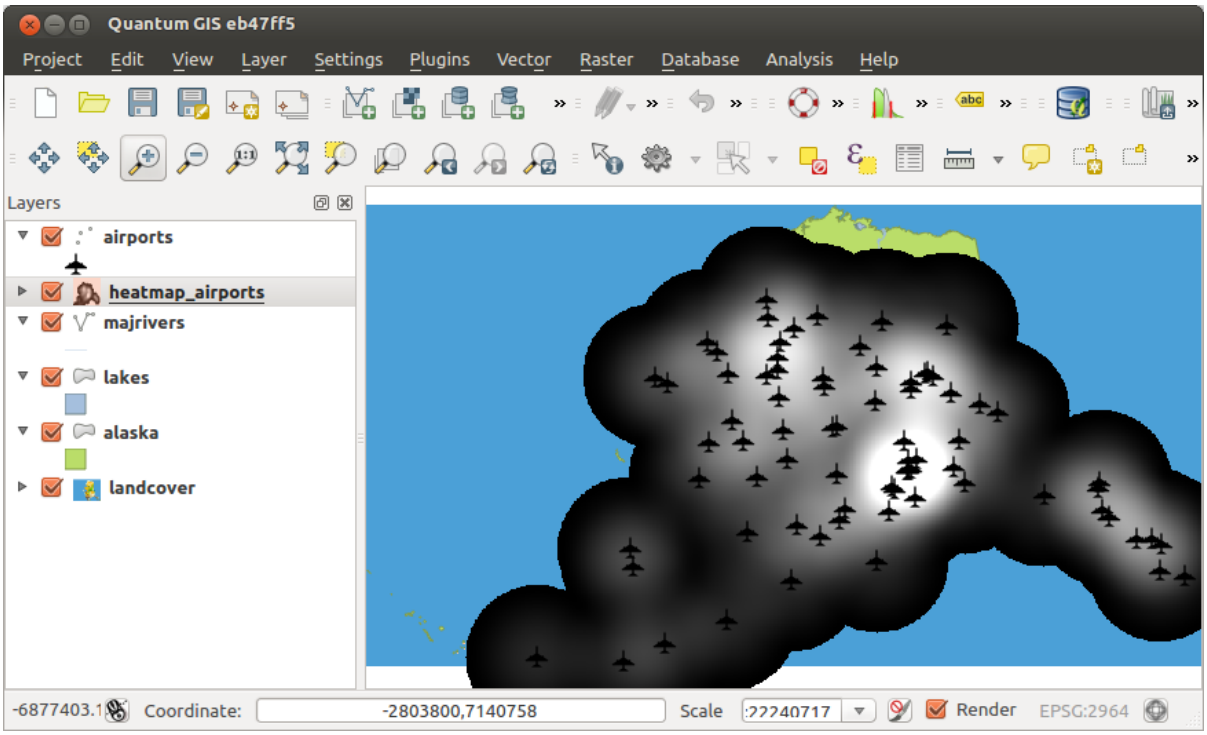


図 27.4: グレーの面に見える読み込み直後のヒートマップ

1. heatmap_airports レイヤのプロパティダイアログを開きます（ heatmap_airports レイヤを選択し、マウス右ボタンでコンテキストメニューを開いて、 **プロパティ** を選択します）。
2. **シンボロジ** タブを選択します。
3. **レンダリングタイプ** を「単バンド疑似カラー」に変更します。
4. 適切な **カラーランプ** 、例えば YlOrRd を選択します。
5. **分類** ボタンをクリックします。
6. *OK* ボタンを押して、レイヤを更新します。

最終的な結果を 図 27.5 に示します。

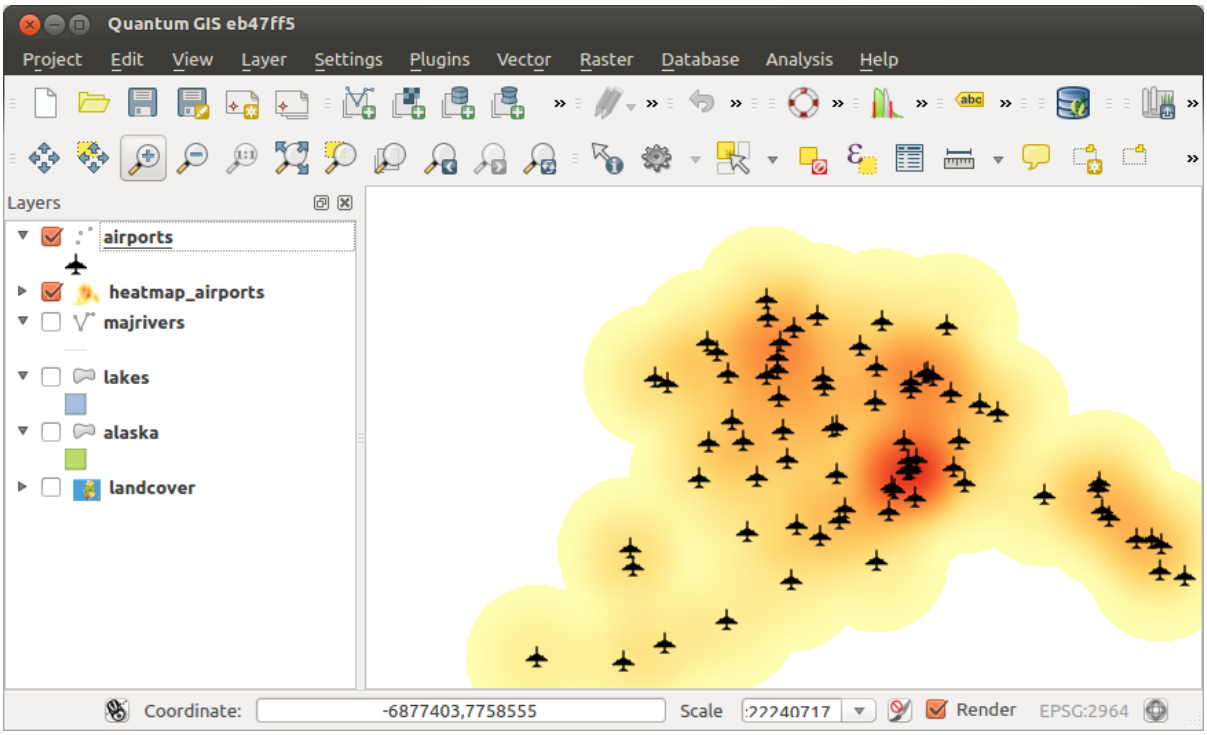


図 27.5: スタイルが設定されたアラスカの空港のヒートマップ

Python コード

Algorithm ID: `qgis:heatmapkerneldensityestimation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

IDW 内挿（逆距離加重法）

ポイントベクタレイヤの逆距離加重法（IDW）内挿ラスタを生成します。

作成したい未知の点からの距離に応じてサンプル点からの影響が減少するように、サンプル点は内挿時に重み付けされます。

IDW 内挿法には、サンプルデータ点の分布が不均一な場合、内挿結果の質が低下するといった、いくつかの欠点もあります。

その上、内挿面の最大値と最小値はサンプルデータ点上でしか起こりえません。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INTERPOLATION_DA	[文字列]	内挿に使用するベクタレイヤとフィールド。コードでは文字列で指定します（詳細については InterpolationWidgets の ParameterInterpolationData クラスを参照）。 内挿データの文字列を構成するために、以下の GUI 要素が用意されています： • **入力ベクタ** [ベクタ：任意] • **内挿対象の属性** [テーブルのフィールド：数値]：内挿に使用する属性 • **内挿に Z 座標を使用する** [ブール値]：レイヤに保存されている Z 値を使用します（デフォルト：False） 追加されたレイヤとフィールドの組み合わせごとに、タイプを選択します： • **ポイント（*Points*）** • **ストラクチャーライン** • **ブレークライン（改行）** 文字列では、複数のレイヤ・フィールド要素は '::|::' で区切られます。レイヤ・フィールド要素のサブ要素は '::~::' で区切られます。
距離係数（P）	DISTANCE_COEFFIC	[数値] デフォルト： 2.0	内挿の距離係数を設定します。最小値は 0.0、最大値は 100.0 です。

次のページに続く

表 27.15 – 前のページからの続き

ラベル	名前	データ型	説明
領域 (xmin, xmax, ymin, ymax)	EXTENT	[範囲]	出力ラスタレイヤの範囲。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算... : 保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
出力ラスタサイズ	PIXEL_SIZE	[数値] デフォルト： 0.1	レイヤの単位で表した、出力ラスタレイヤのピクセルサイズ GUI では、出力ラスタサイズは行数（行）/ 列数（カラム）で指定するか、**または** ピクセルサイズ（ピクセルサイズ `X` / ピクセルサイズ `Y`）で指定します。行数や列数が大きくなると、ピクセルサイズは小さくなり、出力ラスタのファイルサイズは増加します。 行 、 カラム 、 ピクセルサイズ `X` 、 ピクセルサイズ `Y` の値は同時に更新されます。つまり、行数を 2 倍にすれば列数も 2 倍になり、ピクセルサイズは半分になります。出力ラスタの範囲は（ほぼ）変わりません。
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿された値のラスタレイヤ。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	内挿された値のラスタレイヤ

Python コード

Algorithm ID: `qgis:idwinterpolation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線の密度

ラスタの各セルについて、円形状の近傍領域内の線の密度を計算します。この指標は、円形の近傍領域と交差する線分の数をすべて足し合わせ、この合計値をその近傍領域の面積で割ることで得られます。ラインセグメントには、重みづけの係数を適用することができます。

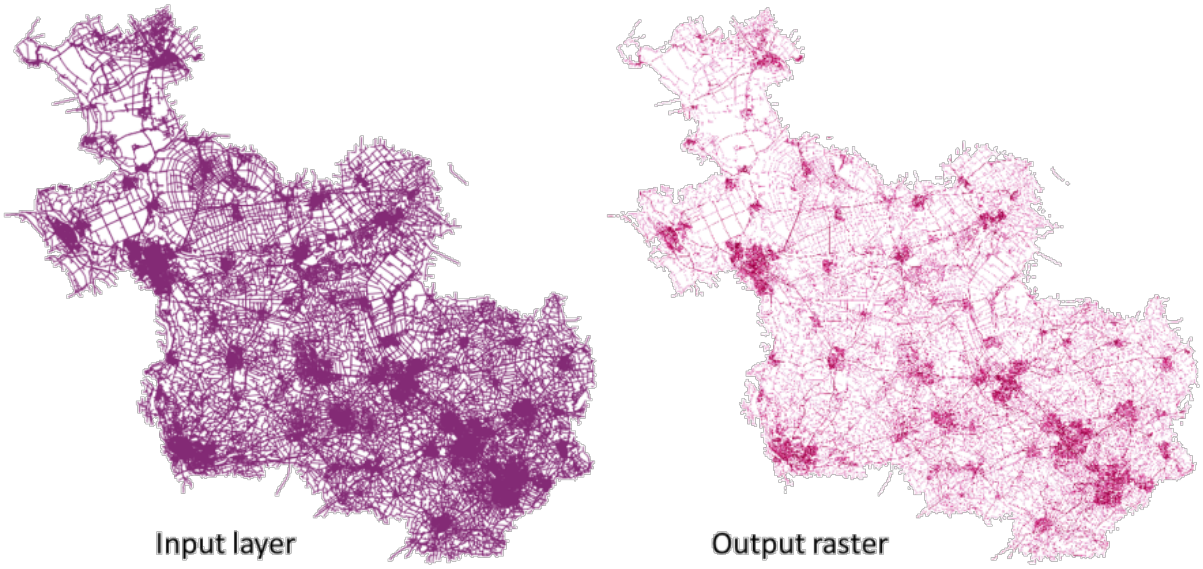


図 27.6: 線の密度の例。ソース入力レイヤ：Roads Overijssel - The Netherlands (OSM)

パラメータ

ラベル	名前	データ型	説明
入力線レイヤ	INPUT	[ベクタ：任意]	ライン地物を含んでいる入力ベクタレイヤ
重み属性（フィールド）	WEIGHT	[数値]	線の密度の計算の際に使用する重み係数が含まれるレイヤのフィールド
検索半径	RADIUS	[数値] デフォルト： 10	円形近傍の検索半径。単位を指定できます。
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 10	レイヤの単位で表した、出力ラスタレイヤのピクセルサイズ。ラスタのピクセルは正方形です。
線密度ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力結果のラスタレイヤ。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
線密度ラスタ	OUTPUT	[ラスタ]	線密度の計算結果のラスタレイヤ

Python コード

Algorithm ID: `native:linedensity`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

TIN 内挿（不規則三角網）

ポイントベクタレイヤの不規則三角網（TIN）内挿ラスタを生成します。

TIN 内挿法を使用して、最近傍点の三角形で形成されるサーフェスを作成できます。このサーフェスを作成するために、選択されたサンプル点の周りの外接円を作成し、その交点を、重ならない、できるだけコンパクトな三角形のネットワークに連結します。結果として得られるサーフェスは滑らかではありません。

このアルゴリズムは、内挿値のラスタレイヤと三角網のベクタラインレイヤの両方を作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	`INTERPOLATION_DA`	[文字列]	内挿に使用するベクタレイヤとフィールド。コードでは文字列で指定します（詳細については InterpolationWidgets の `ParameterInterpolationData` クラスを参照）。 内挿データの文字列を構成するために、以下の GUI 要素が用意されています： • **入力ベクタ** [ベクタ：任意] • **内挿対象の属性** [テーブルのフィールド：数値]：内挿に使用する属性 • **内挿に Z 座標を使用する** [ブール値]：レイヤに保存されている Z 値を使用します（デフォルト：False） 追加されたレイヤとフィールドの組み合わせごとに、タイプを選択します： • **ポイント（*Points*）** • **ストラクチャーライン** • **ブレークライン（改行）** 文字列では、複数のレイヤ・フィールド要素は `'::|::'` で区切られます。レイヤ・フィールド要素のサブ要素は `'::~::'` で区切られます。
内挿方法	`METHOD`	[列挙型] デフォルト： 0	使用する内挿方法を設定します。以下のいずれかです： • **線形** • *Clough-Toucher*（**キュービック法**）

次のページに続く

表 27.19 – 前のページからの続き

ラベル	名前	データ型	説明
領域 (xmin, xmax, ymin, ymax)	EXTENT	[範囲]	出力ラスタレイヤの範囲。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
出力ラスタサイズ	PIXEL_SIZE	[数値] デフォルト： 0.1	レイヤの単位で表した、出力ラスタレイヤのピクセルサイズ GUI では、出力ラスタサイズは行数（行）/ 列数（カラム）で指定するか、**または** ピクセルサイズ（ピクセルサイズ `X` / ピクセルサイズ `Y`）で指定します。行数や列数が大きくなると、ピクセルサイズは小さくなり、出力ラスタのファイルサイズは増加します。 行 、 カラム 、 ピクセルサイズ `X` 、 ピクセルサイズ `Y` の値は同時に更新されます。つまり、行数を 2 倍にすれば列数も 2 倍になり、ピクセルサイズは半分になります。出力ラスタの範囲は（ほぼ）変わりません。
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	TIN 内挿結果のラスタレイヤ。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

次のページに続く

表 27.19 – 前のページからの続き

ラベル	名前	データ型	説明
TIN 内挿出力	TRIANGULATION	[ベクタ：ライン] デフォルト：[出力をスキップ]	TIN 生成結果のベクタレイヤ。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	TIN 内挿結果のラスタレイヤ
TIN 内挿出力	TRIANGULATION	[ベクタ：ライン]	TIN 生成結果のベクタレイヤ

Python コード

Algorithm ID: `qgis:tininterpolation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.6 レイヤツール

レイヤ情報を出力

選択したレイヤの範囲に対応した地物を持つポリゴンレイヤを作ります。

レイヤの詳細（CRS、提供者名、ファイルパス、レイヤ名、サブセットフィルタ、要約、属性）は、各地物の属性として添付されます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	LAYERS	[ベクタ：任意] [リスト]	情報を得たい入力ベクタレイヤ
出力	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	情報を伴った出力レイヤの指定。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力	OUTPUT	[ベクタ：ポリゴン]	入力レイヤの範囲と関連情報を属性で示すポリゴンベクタレイヤ。

Python コード

Algorithm ID: native:exportlayersinformation

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

スプレッドシートへ出力

選択したベクタレイヤの属性をスプレッドシート文書にエクスポートしたり、オプションで既存のスプレッドシートに追加シートとして加えたりします。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	LAYERS	[ベクタ：任意] [リスト]	入力ベクタレイヤ。出力スプレッドシートは、レイヤごとに、このレイヤの属性を含むシートで構成されます。
属性名をカラム名に使用	USE_ALIAS	[ブール値] デフォルト：False	スプレッドシートに属性テーブルのフィールドの別名を使用する。
フォーマットした値を出力	FORMATTED_VALUES	[ブール値] デフォルト：False	もし True なら、整形された人間が読める値 (例えば *value map* や *value relation* から) をスプレッドシートにエクスポートする。
上書きする	OVERWRITE	[ブール値] デフォルト： True	指定したスプレッドシートが存在する場合、このオプションを True に設定すると、その既存のスプレッドシートが上書きされます。このオプションが False でスプレッドシートが存在する場合、レイヤは追加シートとして加えられます。
出力先	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	各レイヤのシートを持った出力スプレッドシート。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力先	OUTPUT	[ファイル]	各レイヤのシートを持ったスプレッドシート。
スプレッドシート内のレイヤ	OUTPUT_LAYERS	[リスト]	スプレッドシートに加えられたシートの一覧。

Python コード

Algorithm ID: `native:exporttospreadsheet`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤ範囲を抽出

入力地物の全てを覆う最小のバウンディングボックス（南北方向の長方形）のベクタレイヤを生成します。

出力レイヤに含まれるのは、入力レイヤ全体に対するただ一つのバウンディングボックスです。

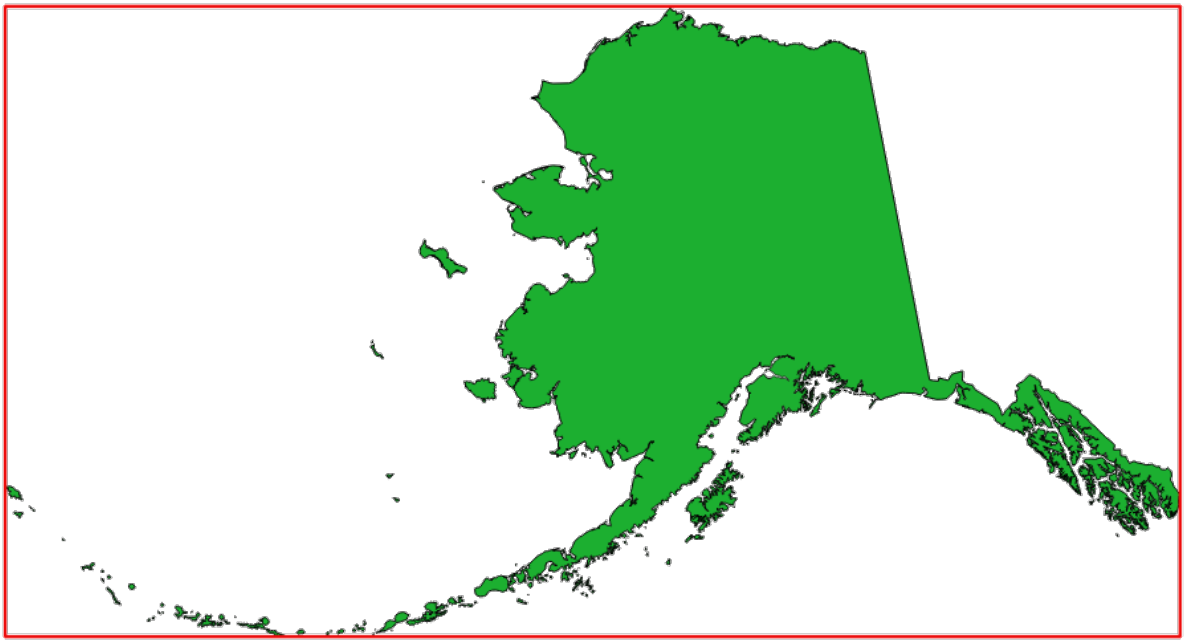


図 27.7: ソースレイヤのバウンディングボックス（赤色の線）

デフォルトメニュー ： ベクタ　 調査ツール

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[レイヤ]	入力レイヤ
領域	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	結果の領域ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
領域	OUTPUT	[ベクタ：ポリゴン]	範囲（最小バウンディングボックス）の出力（ポリゴン）ベクタレイヤ

Python コード

Algorithm ID: qgis:polygonfromlayerextent

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.7 メッシュ

等高線を出力

メッシュのスカラデータセットから等高線をベクタレイヤとして作ります

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	INPUT	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	DATASET_GROUPS	[レイヤ][リスト]	データセットグループ
データセット時間	DATASET_TIME	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
範囲内でインクリメントする オプション	INCREMENT	[数値] デフォルト：**未設定**	生成された水平面の間隔。
最小値 オプション	MINIMUM	[数値] デフォルト：**未設定**	等高線の開始値
最大値 オプション	MAXIMUM	[数値] デフォルト：**未設定**	等高線の最大値、つまり、この値より大きい値は生成されない
等高線の水準値のリスト オプション	CONTOUR_LEVEL_LI	[数値] デフォルト：**未設定**	求める等高線の値のリスト（カンマ区切り）。入力されている場合、増分、最小、および最大フィールドは考慮されない。
出力座標系 オプション	CRS_OUTPUT	[crs]	出力に割り当てる座標参照系
出力の等高線	OUTPUT_LINES	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	メッシュレイヤの等高線を表す出力ラインレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

次のページに続く

表 27.21 – 前のページからの続き

ラベル	名前	データ型	説明
出力の等高線（ポリゴン）	OUTPUT_POLYGONS	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	メッシュレイヤの等高線を表す出力ポリゴンレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力の等高線	OUTPUT_LINES	[ベクタ：ライン]	メッシュレイヤの等高線を表すラインレイヤ
出力の等高線（ポリゴン）	OUTPUT_POLYGONS	[ベクタ：ポリゴン]	メッシュレイヤの等高線を表すポリゴンレイヤ

Python コード

Algorithm ID: `native:meshcontours`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線上のメッシュデータを出力

ベクタレイヤに含まれている線からメッシュデータセットの値を抽出します

各線は、その頂点の値を抽出するための解像度距離パラメータで離散化されます。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	INPUT	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	DATASET_GROUPS	[レイヤ][リスト]	データセットグループ
データセット時間	DATASET_TIME	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
線レイヤ	INPUT_LINES	[ベクタ：ライン]	データセットメッシュからデータを抽出する行
線のセグメント解像度	RESOLUTION	[数値] デフォルト： 10.0	データセットメッシュからデータを抽出するライン上の点間の距離。
データ値の桁数	DATASET_DIGITS	[数値] デフォルト： 2	データセット値の丸め桁数
データを CSV ファイルで出力	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
データを CSV ファイルで出力	OUTPUT	[ファイル]	

Python コード

Algorithm ID: `native:meshexportcrosssection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メッシュの辺を出力

メッシュレイヤの辺を、ラインベクタレイヤに出力します。辺のデータセット値は属性値となります。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	INPUT	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	DATASET_GROUPS	[レイヤ][リスト]	データセットグループ
データセット時間	DATASET_TIME	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
出力座標系 オプション	CRS_OUTPUT	[crs]	出力に割り当てる座標参照系
ベクタオプションを出力	VECTOR_OPTION	[列挙型]	出力するベクタ値の座標の種類。 • 0 -- デカルト座標 (x,y) • 1 -- 極座標 • 2 -- デカルト座標と極座標
出力ベクタレイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ファイルを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力ベクタレイヤ	OUTPUT	[ベクタ：ライン]	入力メッシュレイヤの辺と関連するデータセット値を含んだ出力ラインベクタレイヤ

Python コード

Algorithm ID: `native:exportmeshedges`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メッシュの面を出力

メッシュレイヤの平面を、ポリゴンベクタレイヤに出力します。平面のデータセット値は属性値となります。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	`INPUT`	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	`DATASET_GROUPS`	[レイヤ][リスト]	データセットグループ
データセット時間	`DATASET_TIME`	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
出力座標系 オプション	`CRS_OUTPUT`	[crs]	出力に割り当てる座標参照系
ベクタオプションを出力	`VECTOR_OPTION`	[列挙型]	出力するベクタ値の座標の種類。 • 0 -- デカルト座標 (x,y) • 1 -- 極座標 • 2 -- デカルト座標と極座標
出力ベクタレイヤ	`OUTPUT`	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ファイルを指定します。次のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力ベクタレイヤ	OUTPUT	[ベクタ：ポリゴン]	入力メッシュレイヤの平面と関連するデータセット値を含んだ出力ポリゴンベクタレイヤ

Python コード

Algorithm ID: native:exportmeshfaces

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッドメッシュを出力

メッシュレイヤのデータセット値を、グリッド点ベクタレイヤに出力します。点のデータセット値は属性値になります。

ボリューム上のデータ（3D スタックデータセット値）の場合、出力されたデータセット値は **メッシュレイヤのプロパティ** で定義された方法（デフォルトは Multi level averaging method）を用いて面上で平均化されます。1D メッシュはサポートされていません。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	INPUT	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	DATASET_GROUPS	[レイヤ][リスト]	データセットグループ
データセット時間	DATASET_TIME	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間

次のページに続く

表 27.25 – 前のページからの続き

ラベル	名前	データ型	説明
領域 オプション	EXTENT	[範囲]	データを処理する空間範囲を指定します。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
グリッド間隔 オプション	GRID_SPACING	[数値] デフォルト： 10.0	使用するサンプル点の間隔
出力座標系 オプション	CRS_OUTPUT	[crs]	出力に割り当てる座標参照系
ベクタオプションを出力	VECTOR_OPTION	[列挙型]	出力するベクタ値の座標の種類。 • 0 -- デカルト座標 (x,y) • 1 -- 極座標 • 2 -- デカルト座標と極座標
出力ベクタレイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ファイルを指定します。次のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力ベクタレイヤ	OUTPUT	[ベクタ：ポイント]	オーバーレイ平面から計算されたデータセット値を持つ出力点ベクタレイヤ。

Python コード

Algorithm ID: native:exportmeshongrid

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メッシュの頂点を出力

メッシュレイヤの頂点を、点ベクタレイヤに出力します。頂点のデータセット値は属性値となります。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	INPUT	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	DATASET_GROUPS	[レイヤ][リスト]	データセットグループ
データセット時間	DATASET_TIME	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
出力座標系 オプション	CRS_OUTPUT	[crs]	出力に割り当てる座標参照系
ベクタオプションを出力	VECTOR_OPTION	[列挙型]	出力するベクタ値の座標の種類。 • 0 -- デカルト座標 (x,y) • 1 -- 極座標 • 2 -- デカルト座標と極座標

次のページに続く

表 27.26 – 前のページからの続き

ラベル	名前	データ型	説明
出力ベクタレイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ファイルを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力ベクタレイヤ	OUTPUT	[ベクタ：ポイント]	入力メッシュレイヤの頂点と関連するデータセット値を含む、出力点ベクタレイヤ

Python コード

Algorithm ID: `native:exportmeshvertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メッシュの時系列データ値を出力

ベクタレイヤに含まれる点からメッシュデータセットの時系列値を抽出します。

時間間隔をデフォルト値（0 時間）のままにしておくと、最初に選択されたデータセット・グループの最初の 2 つのデータセットの時間間隔が使われます。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	INPUT	[mesh]	データを抽出するメッシュレイヤ
データセットグループ	DATASET_GROUPS	[レイヤ][リスト]	データセットグループ
開始時刻	STARTING_TIME	[日付時刻]	考慮する時間範囲の開始位置 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
終了時刻	FINISHING_TIME	[日付時刻]	考慮する時間範囲の終了位置 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間
間隔（単位・時） オプション	TIME_STEP	[数値] デフォルト： 0	抽出する 2 つの連続したステップ間の時間。最初に選択したデータセットグループの時間間隔を使用する場合は `0` のままにします。
データを出力する点	INPUT_POINTS	[ベクタ：ポイント]	データセットメッシュからデータを抽出する点を含むベクタレイヤ
座標値の桁数	COORDINATES_DIGI	[数値]	座標値の丸め桁数 デフォルト： 2
データ値の桁数	DATASET_DIGITS	[数値] デフォルト： 2	データセット値の丸め桁数
データを CSV ファイルで出力	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
データを CSV ファイルで出力	OUTPUT	[ファイル]	オーバーレイする点地物のメッシュデータセットの時系列値を含む `.CSV` ファイル

Python コード

Algorithm ID: `native:meshexporttimeseries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メッシュデータをラスタライズ

メッシュデータセットからラスタレイヤを作ります。

ボリューム上のデータ（3D スタックデータセット値）の場合、出力されたデータセット値は **メッシュレイヤのプロパティ** で定義された方法（デフォルトは Multi level averaging method）を用いて面上で平均化されます。1D メッシュはサポートされていません。

パラメータ

ラベル	名前	データ型	説明
入力メッシュ	`INPUT`	[mesh]	データを取り出すメッシュレイヤ
データセットグループ	`DATASET_GROUPS`	[レイヤ][リスト]	データセットグループ
データセット時間	`DATASET_TIME`	[日付時刻]	考慮する時間範囲 • 0 -- 現在のキャンバス時間 • 1 -- 指定された日付 • 2 -- データセットグループの時間

次のページに続く

表 27.28 – 前のページからの続き

ラベル	名前	データ型	説明
領域 オプション	EXTENT	[範囲]	データを処理する空間範囲を指定します。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 1.0	出力ラスタレイヤのピクセルのサイズ。
出力座標系 オプション	CRS_OUTPUT	[crs]	出力に割り当てる座標参照系
出力ラスタレイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタレイヤ	OUTPUT	[ラスタ]	メッシュレイヤから計算されたデータセット値を持つ出力ラスタレイヤ。

Python コード

Algorithm ID: `native:meshrasterize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

TIN メッシュを作成

ベクタレイヤから TIN メッシュレイヤを作ります。TIN メッシュはドロネー三角形分割を使って作られます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	SOURCE_DATA	[ベクタ：任意] [リスト]	メッシュレイヤを生成するために結合するベクタレイヤ
ベクタレイヤ	GUI のみ	[ベクタ：任意] [リスト]	メッシュレイヤを生成するために結合するベクタレイヤのセレクタ
点の値	GUI のみ	[テーブルのフィールド：任意]	選択されたレイヤから使用するフィールドのセレクタ。各頂点には、元の地物の対応する値が割り当てられます。
点の値に Z 座標を使用する	GUI のみ	[ブール値] デフォルト： False	チェックした場合、ベクタレイヤの点またはポリゴン/ラインの頂点の Z 値が、頂点メッシュレイヤの Z 値の割り当てに使用されます。入力レイヤが 3D の場合のみ有効です。
出力形式	MESH_FORMAT	[列挙型] デフォルト：2DM	生成したレイヤの出力形式 • 0 --- 2DM • 1 --- SELAFIN • 2 --- PLY • 3 --- Ugrid
出力座標系 オプション	CRS_OUTPUT	[crs]	出力に割り当てる座標参照系

次のページに続く

表 27.29 – 前のページからの続き

ラベル	名前	データ型	説明
出力ファイル	OUTPUT_MESH	[mesh] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ファイル	OUTPUT_MESH	[mesh]	ベクタレイヤから計算されたデータセット値を持つ出力メッシュレイヤ

Python コード

Algorithm ID: `native:tinmeshcreation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.8 モデラーツール

警告: これらのツールは、モデルデザイナーでのみ利用可能です。プロセシングツールボックスでは使用できません。

条件分け

モデルに条件分岐を追加し、式の評価結果に基づいてモデルのパーツを実行することができます。主にツールの依存関係からモデルのフローを制御するために使用します。

パラメータ

ラベル	名前	データ型	説明
フィールド	BRANCH	[文字列]	条件の名前
フィールド	CONDITION	[式]	評価する式

出力

なし

Python コード

Algorithm ID: native:condition

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ディレクトリの作成

ファイルシステム上に新しいディレクトリを作成します。ディレクトリは再帰的に作成され、指定された完全なディレクトリパスを構築するために、必要なすべての親ディレクトリを作成します。ディレクトリがすでに存在する場合は、エラーは発生しません。

パラメータ

ラベル	名前	データ型	説明
ディレクトリパス	PATH	[文字列]	作成するフォルダーのパス

出力

ラベル	名前	データ型	説明
出力	OUTPUT	[フォルダ]	作成されたフォルダ

Python コード

Algorithm ID: native:createdirectory

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地物フィルタ

入力レイヤの地物をフィルタし、ひとつ又は複数の出力にリダイレクトします。すべての入力レイヤに共通する属性名を知らない場合は、地物ジオメトリと $id や uuid などの一般的なレコードメカニズムでのみフィルタリングが可能です。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力レイヤ
出力とフィルタ（ひとつ以上）	OUTPUT_<name of the filter>	[入力レイヤと同じ]	フィルタと出力レイヤと（フィルタの数だけ）。

出力

ラベル	名前	データ型	説明
出力（ひとつ以上）	native:filter_1: of filter>	[入力レイヤと同じ]	フィルタリングされた地物を持つ出力レイヤ（フィルタの数だけ）。

Python コード

Algorithm ID: `native:filter`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

型によるフィルタ

レイヤを型によってフィルタします。入力されたレイヤーは、ベクタレイヤかラスタレイヤかによって異なる出力に向けられます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[レイヤ]	汎用マップレイヤ

出力

ラベル	名前	データ型	説明
ベクタ地物 オプション	VECTOR	[ベクタ]	互換性のある、入力のベクタレイヤ
ラスタレイヤ オプション	RASTER	[ラスタ]	互換性のある、入力のラスタレイヤ

Python コード

Algorithm ID: `native:filterlayersbytype`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤをプロジェクトに読み込む

レイヤを現在のプロジェクトに読み込みます。

パラメータ

ラベル	名前	データ型	説明
レイヤ	INPUT	[レイヤ]	凡例に読み込むレイヤ
ロードされたレイヤ名	NAME	[文字列]	読み込んだレイヤの名前

出力

ラベル	名前	データ型	説明
レイヤ	OUTPUT	[入力レイヤと同じ]	読み込まれ（て名前を変更され）たレイヤ

Python コード

Algorithm ID: `native:loadlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

例外発生

例外を発生させ、モデルの実行を中止します。例外メッセージはカスタマイズ可能で、オプションとして式による条件も指定できます。条件式を指定する場合、式の結果が True である場合にのみ、例外を発生させます。False の場合には例外は発生せず、モデルの実行が継続します。

パラメータ

ラベル	名前	データ型	説明
エラーメッセージ	MESSAGE	[文字列]	表示するメッセージ
条件式 オプション	CONDITION	[式]	評価する式

出力

ログパネルのメッセージ。

Python コード

Algorithm ID: native:raiseexception

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

メッセージを表示

NEW in 3.26

ログに情報メッセージを発生します。メッセージはカスタマイズ可能で、オプションとして式による条件も指定できます。条件式を指定する場合、式の結果が true である場合にのみメッセージがログに記録されます。結果が false の場合、メッセージは記録されません。

パラメータ

ラベル	名前	データ型	説明
情報メッセージ	MESSAGE	[文字列]	表示するメッセージ
条件式 オプション	CONDITION	[式]	評価する式

出力

ログパネルのメッセージ。

Python コード

Algorithm ID: `native:raisemessage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

警告メッセージ出力

ログに警告メッセージを表示します。この警告メッセージはカスタマイズ可能で、オプションとして式による条件を指定できます。条件式を指定する場合、条件式が True である場合にのみ、警告メッセージを表示します。False の場合には警告は表示されません。

パラメータ

ラベル	名前	データ型	説明
エラーメッセージ	`MESSAGE`	[文字列]	表示するメッセージ
条件式	`CONDITION`	[式]	評価する式
オプション			

出力

ログパネルのメッセージ。

Python コード

Algorithm ID: `native:raisewarning`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤ名の変更

レイヤ名を変更します。

パラメータ

ラベル	名前	データ型	説明
レイヤ	INPUT	[レイヤ]	名前を変更したいレイヤ
新しい名前	NAME	[文字列]	レイヤの新しい名前

出力

ラベル	名前	データ型	説明
レイヤ	OUTPUT	[入力レイヤと同じ]	（名前を変更された）結果のレイヤ

Python コード

Algorithm ID: `native:renamelayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ログをファイルに保存する

モデルの実行ログをファイルに保存します。オプションとして、HMTL フォーマットでログを出力できます。

パラメータ

ラベル	名前	データ型	説明
HTML フォーマットを使う	USE_HTML	[ブール値] デフォルト：False	ログに HTML 形式を使用します

出力

ラベル	名前	データ型	説明
ログファイル	OUTPUT	[文字列]	ログの保存先

Python コード

Algorithm ID: `native:savelog`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

プロジェクト変数を設定

現在のプロジェクトの式変数を設定します。

パラメータ

ラベル	名前	データ型	説明
変数名	NAME	[文字列]	変数の名前
変数値	VALUE	[文字列]	格納される値

出力

なし

Python コード

Algorithm ID: native:setprojectvariable

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

文字列連結

プロセシングモデラー内で、2 つの文字列を 1 つに連結します。

パラメータ

ラベル	名前	データ型	説明
入力 1	INPUT_1	[文字列]	1 つ目の文字列
入力 2	INPUT_2	[文字列]	2 つめの文字列

出力

ラベル	名前	データ型	説明
連結文字列	CONCATENATION	[文字列]	連結された文字列

Python コード

Algorithm ID: native:stringconcatenation

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

可変距離バッファ

警告: このアルゴリズムは非推奨です。代わりに **バッファ（*buffer*）** アルゴリズムを使用してください。

入力レイヤにある全ての地物のバッファ領域を計算します。

与えられた地物のバッファサイズは属性で定義されるため、異なる地物が異なるバッファサイズを持つことができます。

参考:

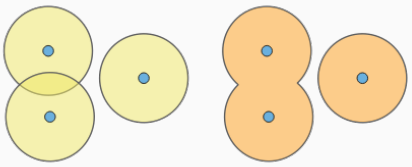
バッファ（*buffer*）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
距離フィールド	DISTANCE	[テーブルのフィールド：数値]	バッファの距離半径の属性
セグメント	SEGMENTS	[数値] デフォルト: *5*	丸みを帯びたオフセットの四分円を近似するために使用するセグメントの数を制御します

次のページに続く

表 27.30 – 前のページからの続き

ラベル	名前	データ型	説明
結果を融合する	DISSOLVE	[ブール値] デフォルト: *False*	融合する最終的なバッファを選び、全ての入力地物をカバーする単独の地物にします。  図 27.8: 通常と融合したバッファ
線端スタイル	END_CAP_STYLE	[列挙型] デフォルト: Round	バッファでラインの末端をどのように扱うかを制御します。  図 27.9: Round、flat、square の線端スタイル
継ぎ目スタイル	JOIN_STYLE	[列挙型] デフォルト: Round	ライン内の角をオフセットする際に、Round、Miter、Beveled のいずれの結合を使用するかを指定します。
miter 制限	MITER_LIMIT	[数値] デフォルト： 2.0	継ぎ目スタイルが Miter の場合にのみ適用され、Miter 継ぎ目の作成時に使用するオフセット曲線からの最大距離を制御します。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	バッファポリゴンベクタレイヤ。

Python コード

Algorithm ID: `qgis:variabledistancebuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.9 ネットワーク解析

サービスエリア（始点レイヤ）

ポイントレイヤを始点として、ある距離または時間以内に到達可能なネットワークの全ての辺または辺の一部を返します。これにより、例えば、道路ネットワーク上で、指定された値以上のコスト（距離または時間）をかけずに移動できる場所はどこか、というようなネットワーク内のアクセス性を評価できます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ネットワークを表すベクタレイヤ	INPUT	[ベクタ：ライン]	対象となるネットワークを表すラインベクタレイヤ
始点のベクタレイヤ	START_POINTS	[ベクタ：ポイント]	サービスエリア生成のための始点となる地物のポイントベクタレイヤ
計算するパスの種類	STRATEGY	[列挙型] デフォルト：0	計算するパスの種類。次のいずれかです： • 0 --- 最短 • 1 --- 最速
求めたい旅行コスト（最短なら単位は距離、最速なら単位は時間）	TRAVEL_COST	[数値] デフォルト：0	この値は、 **最短** パスを求めたい場合には（ネットワークレイヤの単位での）距離として評価され、 **最速** パスを求めたい場合には時間（単位は時間）として評価されます。

次のページに続く

表 27.31 – 前のページからの続き

ラベル	名前	データ型	説明
サービスエリア（線）	OUTPUT_LINES	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	サービスエリアの出力ラインレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
サービスエリア（境界点）	OUTPUT	[ベクタ：ポイント] デフォルト：[出力をスキップ]	サービスエリアの境界ノードの出力ポイントレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
向きを示す属性（フィールド） オプション	DIRECTION_FIELD	[テーブルのフィールド：文字列] デフォルト： 0.0	ネットワークの辺の方向を指定するためのフィールド。 このフィールドで使用する値は、3 つのパラメータ 順方向を示す値 、 逆方向を示す値、双方向の値 で指定します。「順方向」と「逆方向」は一方通行の辺に対応し、「双方向」は順・逆両方向の辺であることを表します。もしある地物がこのフィールドに値を持っていなかったり、フィールドが設定されていない場合には、デフォルトの方向設定（ デフォルトの方向 パラメータで指定）が使用されます。
順方向を示す値 オプション	VALUE_FORWARD	[文字列] デフォルト：''（空の文字列）	順方向の辺を識別するために方向フィールドに設定される値

次のページに続く

表 27.32 – 前のページからの続き

ラベル	名前	データ型	説明
逆方向を示す値 オプション	VALUE_BACKWARD	[文字列] デフォルト：''（空の文字列）	逆方向の辺を識別するために方向フィールドに設定される値
双方向の値 オプション	VALUE_BOTH	[文字列] デフォルト：''（空の文字列）	双方向の辺を識別するために方向フィールドに設定される値
デフォルトの方向	DEFAULT_DIRECTIO	[列挙型] デフォルト： 2	地物の方向フィールドに値が設定されていないか、方向フィールドが設定されていない場合には、この方向値が使用されます。次のいずれかです： • 0 --- 順方向 • 1 --- 逆方向 • 2 --- 双方向
速度を示す属性（フィールド） オプション	SPEED_FIELD	[テーブルのフィールド：文字列]	最速パスを求める場合に、ネットワークの辺の速度（ km/h 単位）を指定するフィールド 地物が速度フィールドの値を持たないか、速度フィールドが設定されていない場合には、デフォルトの速度値（ デフォルトの速度 パラメータで指定）が使用されます。
デフォルトの速度（km/h）	DEFAULT_SPEED	[数値] デフォルト： 50.0	辺に速度フィールドが指定されていない場合に、移動時間の計算に使用する値
トポロジ許容値（接続しているとみなす点間距離）	TOLERANCE	[数値] デフォルト： 0.0	指定された許容値よりも端点が接近している場合には、2 つのラインは接続しているとみなします

上限下限の点を含む	INCLUDE_BOUNDS	[ブール値] デフォルト： False	サービスエリア境界において、各辺について 2 点ずつのポイントレイヤ出力を作成します。点の 1 つは辺の始点で、2 つ目は終点です。

出力

ラベル	名前	データ型	説明
サービスエリア(境界点)	OUTPUT	[ベクタ：ポイント]	サービスエリアの境界ノードの出力ポイントレイヤ
サービスエリア(線)	OUTPUT_LINES	[ベクタ：ライン]	開始点から指定されたコストで到達可能なネットワークの部分を表すラインレイヤ

Python コード

Algorithm ID: `qgis:serviceareafromlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

サービスエリア（始点を指定）

ポイント t 地物を始点として、ある距離または時間以内に到達可能なネットワークの全ての辺または辺の一部を返します。これにより、例えば、道路ネットワーク上で、指定された値以上のコスト（距離または時間）をかけずに移動できる場所はどこか、というようなネットワーク内のアクセス性を評価できます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ネットワークを表すベクタレイヤ	INPUT	[ベクタ：ライン]	対象となるネットワークを表すラインベクタレイヤ
始点	START_POINT	[座標]	周辺のサービスエリアを計算するポイントの座標
計算するパスの種類	STRATEGY	[列挙型] デフォルト：0	計算するパスの種類。次のいずれかです： • 0 --- 最短 • 1 --- 最速

次のページに続く

表 27.33 – 前のページからの続き

ラベル	名前	データ型	説明
求めたい旅行コスト（最短なら単位は距離、最速なら単位は時間）	TRAVEL_COST	[数値] デフォルト： 0	この値は、 **最短** パスを求めたい場合には（ネットワークレイヤの単位での）距離として評価され、 **最速** パスを求めたい場合には時間（単位は時間）として評価されます。
サービスエリア（線）	OUTPUT_LINES	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	サービスエリアの出力ラインレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
サービスエリア（境界点）	OUTPUT	[ベクタ：ポイント] デフォルト：[出力をスキップ]	サービスエリアの境界ノードの出力ポイントレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
向きを示す属性（フィールド） オプション	DIRECTION_FIELD	[テーブルのフィールド：文字列] デフォルト： 0.0	ネットワークの辺の方向を指定するためのフィールド。 このフィールドで使用する値は、3 つのパラメータ 順方向を示す値 、 逆方向を示す値 、双方向の値 で指定します。「順方向」と「逆方向」は一方通行の辺に対応し、「双方向」は順・逆両方向の辺であることを表します。もしある地物がこのフィールドに値を持っていなかったり、フィールドが設定されていない場合には、デフォルトの方向設定（ デフォルトの方向 パラメータで指定）が使用されます。
順方向を示す値 オプション	VALUE_FORWARD	[文字列] デフォルト：''（空の文字列）	順方向の辺を識別するために方向フィールドに設定される値
逆方向を示す値 オプション	VALUE_BACKWARD	[文字列] デフォルト：''（空の文字列）	逆方向の辺を識別するために方向フィールドに設定される値
双方向の値 オプション	VALUE_BOTH	[文字列] デフォルト：''（空の文字列）	双方向の辺を識別するために方向フィールドに設定される値
デフォルトの方向	DEFAULT_DIRECTIO	[列挙型] デフォルト： 2	地物の方向フィールドに値が設定されていないか、方向フィールドが設定されていない場合には、この方向値が使用されます。次のいずれかです： • 0 --- 順方向 • 1 --- 逆方向 • 2 --- 双方向
速度を示す属性（フィールド） オプション	SPEED_FIELD	[テーブルのフィールド：文字列]	最速パスを求める場合に、ネットワークの辺の速度（ km/h 単位）を指定するフィールド 地物が速度フィールドの値を持たないか、速度フィールドが設定されていない場合には、デフォルトの速度値（ デフォルトの速度 パラメータで指定）が使用されます。
デフォルトの速度（km/h）	DEFAULT_SPEED	[数値] デフォルト： 50.0	辺に速度フィールドが指定されていない場合に、移動時間の計算に使用する値
トポロジ許容値（接続しているとみなす点間距離）	TOLERANCE	[数値] デフォルト： 0.0	指定された許容値よりも端点が接近している場合には、2 つのラインは接続しているとみなします

上限下限の点を含む	INCLUDE_BOUNDS	[ブール値] デフォルト：False	サービスエリア境界において、各辺について 2 点ずつのポイントレイヤ出力を作成します。点の 1 つは辺の始点で、2 つ目は終点です。

出力

ラベル	名前	データ型	説明
サービスエリア（境界点）	OUTPUT	[ベクタ：ポイント]	サービスエリアの境界ノードの出力ポイントレイヤ
サービスエリア（線）	OUTPUT_LINES	[ベクタ：ライン]	開始点から指定されたコストで到達可能なネットワークの部分を表すラインレイヤ

Python コード

Algorithm ID: `native:serviceareafrompoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最短経路（始点レイヤから指定終点）

ベクタレイヤで定義された複数の始点から指定した終点までの最適経路（最短または最速）を計算します。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ネットワークを表すベクタレイヤ	INPUT	[ベクタ：ライン]	対象となるネットワークを表すラインベクタレイヤ

次のページに続く

表 27.35 – 前のページからの続き

ラベル	名前	データ型	説明
計算するパスの種類	STRATEGY	[列挙型] デフォルト： 0	計算するパスの種類。次のいずれかです： • 0 --- 最短 • 1 --- 最速
始点のベクタレイヤ	START_POINTS	[ベクタ：ポイント]	経路の始点として使用する地物のポイントベクタレイヤ
終点	END_POINT	[座標]	経路の終点を表すポイント地物
出力レイヤ	OUTPUT	[ベクタ：ライン]	最短経路の出力ラインレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
向きを示す属性（フィールド） オプション	DIRECTION_FIELD	[テーブルのフィールド：文字列] デフォルト： 0.0	ネットワークの辺の方向を指定するためのフィールド。 このフィールドで使用する値は、3 つのパラメータ 順方向を示す値 、 逆方向を示す値、双方向の値 で指定します。「順方向」と「逆方向」は一方通行の辺に対応し、「双方向」は順・逆両方向の辺であることを表します。もしある地物がこのフィールドに値を持っていなかったり、フィールドが設定されていない場合には、デフォルトの方向設定（ デフォルトの方向 パラメータで指定）が使用されます。
順方向を示す値 オプション	VALUE_FORWARD	[文字列] デフォルト：''（空の文字列）	順方向の辺を識別するために方向フィールドに設定される値
逆方向を示す値 オプション	VALUE_BACKWARD	[文字列] デフォルト：''（空の文字列）	逆方向の辺を識別するために方向フィールドに設定される値

次のページに続く

表 27.36 – 前のページからの続き

ラベル	名前	データ型	説明
双方向の値 オプション	VALUE_BOTH	[文字列] デフォルト：''（空の文字列）	双方向の辺を識別するために方向フィールドに設定される値
デフォルトの方向	DEFAULT_DIRECTIO	[列挙型] デフォルト： 2	地物の方向フィールドに値が設定されていないか、方向フィールドが設定されていない場合には、この方向値が使用されます。次のいずれかです： • 0 --- 順方向 • 1 --- 逆方向 • 2 --- 双方向
速度を示す属性（フィールド） オプション	SPEED_FIELD	[テーブルのフィールド：文字列]	最速パスを求める場合に、ネットワークの辺の速度（ km/h 単位）を指定するフィールド 地物が速度フィールドの値を持たないか、速度フィールドが設定されていない場合には、デフォルトの速度値（ デフォルトの速度 パラメータで指定）が使用されます。
デフォルトの速度（km/h）	DEFAULT_SPEED	[数値] デフォルト： 50.0	辺に速度フィールドが指定されていない場合に、移動時間の計算に使用する値
トポロジ許容値（接続しているとみなす点間距離）	TOLERANCE	[数値] デフォルト： 0.0	指定された許容値よりも端点が接近している場合には、2 つのラインは接続しているとみなします

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	各始点から終点までの最短または最速経路のラインレイヤ

Python コード

Algorithm ID: native:shortestpathlayertopoint

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最短経路（指定始点から終点レイヤ）

指定した始点とポイントベクタレイヤで定義された複数の終点の間の最適経路（最短または最速）を計算します。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ネットワークを表すベクタレイヤ	INPUT	[ベクタ：ライン]	対象となるネットワークを表すラインベクタレイヤ
計算するパスの種類	STRATEGY	[列挙型] デフォルト：0	計算するパスの種類。次のいずれかです： • 0 --- 最短 • 1 --- 最速
始点	START_POINT	[座標]	経路の始点を表すポイント地物
終点のベクタレイヤ	END_POINTS	[ベクタ：ポイント]	経路の終点として使用する地物のポイントベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ライン]	最短経路の出力ラインレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
向きを示す属性（フィールド） オプション	DIRECTION_FIELD	[テーブルのフィールド：文字列] デフォルト： 0.0	ネットワークの辺の方向を指定するためのフィールド。 このフィールドで使用する値は、3 つのパラメータ 順方向を示す値 、 逆方向を示す値 、双方向の値 で指定します。「順方向」と「逆方向」は一方通行の辺に対応し、「双方向」は順・逆両方向の辺であることを表します。もしある地物がこのフィールドに値を持っていなかったり、フィールドが設定されていない場合には、デフォルトの方向設定（ デフォルトの方向 パラメータで指定）が使用されます。
順方向を示す値 オプション	VALUE_FORWARD	[文字列] デフォルト：''（空の文字列）	順方向の辺を識別するために方向フィールドに設定される値
逆方向を示す値 オプション	VALUE_BACKWARD	[文字列] デフォルト：''（空の文字列）	逆方向の辺を識別するために方向フィールドに設定される値
双方向の値 オプション	VALUE_BOTH	[文字列] デフォルト：''（空の文字列）	双方向の辺を識別するために方向フィールドに設定される値
デフォルトの方向	DEFAULT_DIRECTIO	[列挙型] デフォルト： 2	地物の方向フィールドに値が設定されていないか、方向フィールドが設定されていない場合には、この方向値が使用されます。次のいずれかです： • 0 --- 順方向 • 1 --- 逆方向 • 2 --- 双方向
速度を示す属性（フィールド） オプション	SPEED_FIELD	[テーブルのフィールド：文字列]	最速パスを求める場合に、ネットワークの辺の速度（ km/h 単位）を指定するフィールド 地物が速度フィールドの値を持たないか、速度フィールドが設定されていない場合には、デフォルトの速度値（ デフォルトの速度 パラメータで指定）が使用されます。
デフォルトの速度（km/h）	DEFAULT_SPEED	[数値] デフォルト： 50.0	辺に速度フィールドが指定されていない場合に、移動時間の計算に使用する値
トポロジ許容値（接続しているとみなす点間距離）	TOLERANCE	[数値] デフォルト： 0.0	指定された許容値よりも端点が接近している場合には、2 つのラインは接続しているとみなします

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	各始点から終点までの最短または最速経路のラインレイヤ

Python コード

Algorithm ID: native:shortestpathpointtolayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最短経路（指定始点から指定終点）

指定した始点と指定した終点の間の最適経路（最短または最速）を計算します。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ネットワークを表すベクタレイヤ	INPUT	[ベクタ：ライン]	対象となるネットワークを表すラインベクタレイヤ
計算するパスの種類	STRATEGY	[列挙型] デフォルト：0	計算するパスの種類。次のいずれかです： • 0 --- 最短 • 1 --- 最速
始点	START_POINT	[座標]	経路の始点を表すポイント地物
終点	END_POINT	[座標]	経路の終点を表すポイント地物

次のページに続く

表 27.39 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	最短経路の出力ラインレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
向きを示す属性（フィールド） オプション	DIRECTION_FIELD	[テーブルのフィールド：文字列] デフォルト： 0.0	ネットワークの辺の方向を指定するためのフィールド。 このフィールドで使用する値は、3 つのパラメータ 順方向を示す値 、 逆方向を示す値、双方向の値 で指定します。「順方向」と「逆方向」は一方通行の辺に対応し、「双方向」は順・逆両方向の辺であることを表します。もしある地物がこのフィールドに値を持っていなかったり、フィールドが設定されていない場合には、デフォルトの方向設定（ デフォルトの方向 パラメータで指定）が使用されます。
順方向を示す値 オプション	VALUE_FORWARD	[文字列] デフォルト：''（空の文字列）	順方向の辺を識別するために方向フィールドに設定される値
逆方向を示す値 オプション	VALUE_BACKWARD	[文字列] デフォルト：''（空の文字列）	逆方向の辺を識別するために方向フィールドに設定される値
双方向の値 オプション	VALUE_BOTH	[文字列] デフォルト：''（空の文字列）	双方向の辺を識別するために方向フィールドに設定される値

次のページに続く

表 27.40 – 前のページからの続き

ラベル	名前	データ型	説明
デフォルトの方向	DEFAULT_DIRECTIO	[列挙型] デフォルト： 2	地物の方向フィールドに値が設定されていないか、方向フィールドが設定されていない場合には、この方向値が使用されます。次のいずれかです： • 0 --- 順方向 • 1 --- 逆方向 • 2 --- 双方向
速度を示す属性（フィールド） オプション	SPEED_FIELD	[テーブルのフィールド：文字列]	最速パスを求める場合に、ネットワークの辺の速度（ km/h 単位）を指定するフィールド 地物が速度フィールドの値を持たないか、速度フィールドが設定されていない場合には、デフォルトの速度値（ デフォルトの速度 パラメータで指定）が使用されます。
デフォルトの速度（km/h）	DEFAULT_SPEED	[数値] デフォルト： 50.0	辺に速度フィールドが指定されていない場合に、移動時間の計算に使用する値
トポロジ許容値（接続しているとみなす点間距離）	TOLERANCE	[数値] デフォルト： 0.0	指定された許容値よりも端点が接近している場合には、2 つのラインは接続しているとみなします

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	始点から終点までの最短または最速経路のラインレイヤ

Python コード

Algorithm ID: `native:shortestpathpointtopoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.10 プロット

ベクタレイヤの棒グラフ

カテゴリとレイヤの属性値から、棒グラフを作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
カテゴリを示す属性（フィールド）	NAME_FIELD	[テーブルのフィールド：任意]	棒グラフをグループ化するために使用するカテゴリ値のフィールド（X 軸）
値フィールド	VALUE_FIELD	[テーブルのフィールド：任意]	グラフに使用する値（Y 軸）
ベクタレイヤの棒グラフ	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ベクタレイヤの棒グラフ	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: `qgis:barplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤのボックスプロット

カテゴリ属性とレイヤの数値属性から、ボックスプロットを作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
カテゴリを示す属性（フィールド）	NAME_FIELD	[テーブルのフィールド：任意]	ボックスをグループ化するために使用するカテゴリ値のフィールド（X 軸）
値フィールド	VALUE_FIELD	[テーブルのフィールド：任意]	グラフに使用する値（Y 軸）
追加の統計指標	MSD	[列挙型] デフォルト：0	プロットに追加する統計情報。次のいずれかです： • 0 --- 平均を表示 • 1 --- 標準偏差を表示 • 2 --- 平均と標準偏差を表示しない
ベクタレイヤのボックスプロット	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ベクタレイヤのボックスプロット	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: `qgis:boxplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズム**

をコンソールから使う を参照してください。

テーブルの平均・標準偏差のプロット

平均と標準偏差の表示のあるボックスプロットを作成します。

パラメータ

ラベル	名前	データ型	説明
入力テーブル	INPUT	[ベクタ：任意]	入力ベクタレイヤ
カテゴリを示す属性（フィールド）	NAME_FIELD	[テーブルのフィールド：任意]	ボックスをグループ化するために使用するカテゴリ値のフィールド（X 軸）
値フィールド	VALUE_FIELD	[テーブルのフィールド：任意]	グラフに使用する値（Y 軸）
出力のプロット	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力のプロット	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: `qgis:meanandstandarddeviationplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤのポーラープロット

入力ベクタレイヤの値に基づいたポーラープロットを作成します。

パラメータとして 2 つのフィールドを入力する必要があります。1 つは各地物のカテゴリを定義するフィールド（地物をグループ化するため）で、もう 1 つはプロットする値のフィールド（数値でなければならない）です。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
カテゴリを示す属性（フィールド）	NAME_FIELD	[テーブルのフィールド：任意]	地物をグループ化するために使用するカテゴリ値のフィールド（X 軸）
値フィールド	VALUE_FIELD	[テーブルのフィールド：任意]	グラフに使用する値（Y 軸）
ベクタレイヤのポーラープロット	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ベクタレイヤのポーラープロット	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: qgis:polarplot

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタレイヤのヒストグラム

ラスタレイヤの値のヒストグラムを作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド]	ヒストグラムに使用するラスタのバンド番号
階級数	BINS	[数値] デフォルト： 10	ヒストグラムに使用するビンの数（X 軸）。最小値は 2 です。
ヒストグラム	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ヒストグラム	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: `qgis:rasterlayerhistogram`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤのヒストグラム

ベクタレイヤの属性値のヒストグラムを作成します。

ヒストグラムの計算に使用する属性は数値でなければなりません。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
属性	FIELD	[テーブルのフィールド：任意]	グラフに使用する値（Y 軸）
階級数	BINS	[数値] デフォルト： 10	ヒストグラムに使用するビンの数（X 軸）。最小値は 2 です。
ヒストグラム	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ヒストグラム	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: `qgis:vectorlayerhistogram`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤの散布図

ベクタレイヤのシンプルな X - Y 散布図を作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
X 属性値	XFIELD	[テーブルのフィールド：任意]	X 軸に使用するフィールド
Y 属性値	YFIELD	[テーブルのフィールド：任意]	Y 軸に使用するフィールド
散布図の出力	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
散布図の出力	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: qgis:vectorlayerscatterplot

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤの 3D 散布図

ベクタレイヤの 3D 散布図を作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
X 属性値	XFIELD	[テーブルのフィールド：任意]	X 軸に使用するフィールド
Y 属性値	YFIELD	[テーブルのフィールド：任意]	Y 軸に使用するフィールド
Z 属性値	ZFIELD	[テーブルのフィールド：任意]	Z 軸に使用するフィールド
ヒストグラム	OUTPUT	[html] デフォルト：[一時ファイルに保存]	グラフの出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
ヒストグラム	OUTPUT	[html]	グラフの HTML ファイル。 **プロセシング　結果ビューア** からアクセスできます。

Python コード

アルゴリズム ID: `qgis:scatter3dplot`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.11 ラスタ分析

ラスタ値のパーセントランク

単独の入力値に基づいて、ラスタのスタックのセル単位のパーセントランク値を計算し、出力ラスタに書き込みます。

各セル位置で指定された値は、入力ラスタのセル値を重ねてソートしてできたスタックのそれぞれの数値群の中で順位付けされます。スタック値の分布から外れた値については、その値をセル値群の中で順位付けすることができないため、アルゴリズムは NoData を返します。

百分位数の計算には二つの方法があります:

- 線形内挿 (PERCENTRANK.INC)
- 線形内挿 (PERCENTRANK.EXC)

線形補間法は、異なる値にユニークなパーセントランクを返します。どちらの補間法も、LibreOffice や Microsoft Excel で実装されている同様の方法に準じています。

出力ラスタの範囲と解像度は、参照ラスタによって決まります。参照ラスタレイヤのセルサイズと一致しない入力ラスタレイヤは、最近傍再サンプリングを使用して再サンプリングされます。「NoData を無視」パラメータが設定されていない場合、入力レイヤのいずれかに NoData 値があると、NoData セル出力になります。出力されるラスタのデータ型は常に `Float32` となります。

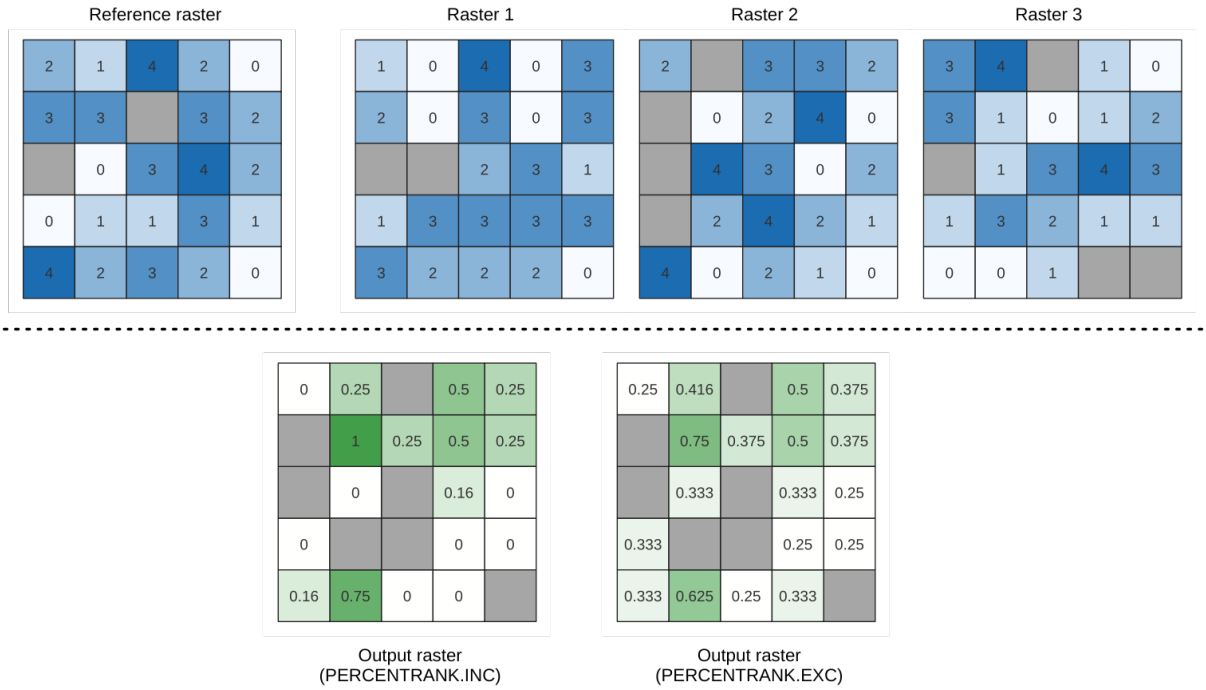


図 27.10: パーセントランク値 = 1。`NoData` セル（灰色）は無視されます。

参考:

セルスタックの百分位数, セルスタック：ラスタ値のパーセントランク

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	評価したいラスタレイヤ群。データラスタスタックにマルチバンドラスタを使用する場合には、このアルゴリズムは常にラスタの最初のバンドに対して解析を行う
方法	METHOD	[列挙型] デフォルト： 0	百分位数の計算の方法 • 0 --- 線形内挿 (PERCENTRANK.INC) • 1 --- 線形内挿 (PERCENTRANK.EXC)
値	VALUE	[数値] デフォルト： 10.0	入力ラスタのセル値を重ねてソートしてできたスタックのそれぞれの数値群の中で順位付けする数値
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト： True	チェックしないとき、入力レイヤの NoData セルは出力レイヤの NoData セルとなります
スナップで参照するレイヤ	REFERENCE_LAYER	[ラスタ]	出力レイヤの作成に使用する参照レイヤ（範囲、CRS、ピクセルの大きさ）
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	OUTPUT_NODATA_VA	[数値] デフォルト： -9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: `native:cellstackpercentrankfromvalue`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

セルスタックの百分位数

ラスタのスタックの、セル単位のパーセンタイル値を計算し、その結果を出力ラスタに書き込みます。返すパーセンタイルは、パーセンタイル入力値（0 から 1 までの範囲）によって決まります。各セル位置で、入力ラスタのセル値を重ねてソートしてできたスタックからそれぞれの値を使って、指定されたパーセンタイルを求めます。

パーセンタイルの計算には三つの方法があります:

- 最近傍ランク: 指定されたパーセンタイル値に最も近い値を返します
- 線形内挿 (PERCENTRANK.INC)
- 線形内挿 (PERCENTRANK.EXC)

線形補間法は、異なるパーセンタイルにユニークな値を返します。どちらの補間法も LibreOffice や Microsoft Excel で実装されているそれぞれの方法に従っています。

出力ラスタの範囲と解像度は、参照ラスタによって決まります。参照ラスタレイヤのセルサイズと一致しない入力ラスタレイヤは、最近傍再サンプリングを使用して再サンプリングされます。「NoData を無視」パラメータが設定されていない場合、入力レイヤのいずれかに NoData 値があると、NoData セル出力になります。出力されるラスタのデータ型は常に `Float32` となります。

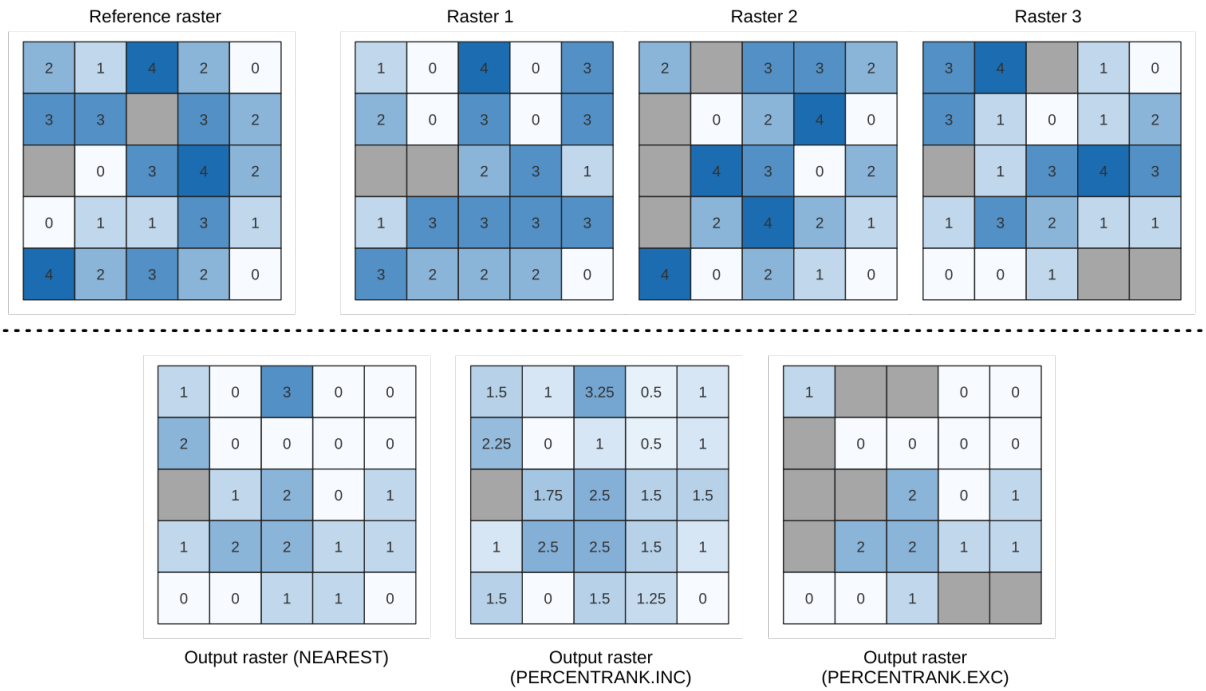


図 27.11: パーセンタイル = 0.25. NoData セル（灰色）は無視されます。

参考:

セルスタックの百分位数, **セルスタック：ラスタ値のパーセントランク**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	評価したいラスタレイヤ群。データラスタスタックにマルチバンドラスタを使用する場合には、このアルゴリズムは常にラスタの最初のバンドに対して解析を行う
方法	METHOD	[列挙型] デフォルト： 0	百分位数の計算の方法 • 0 -- 最近傍ランク: 指定されたパーセンタイルに最も近い値を返します • 1 -- 線形内挿 (PERCENTILE.INC) • 2 -- 線形内挿 (PERCENTILE.EXC)
百分位数	VALUE	[数値] デフォルト： 0.25	入力ラスタのセル値を重ね、ソートしてできたスタックのそれぞれの数値群の中で順位付けした数値。0 と 1 の間。

次のページに続く

表 27.44 – 前のページからの続き

ラベル	名前	データ型	説明
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト： True	チェックしないとき、入力レイヤの No-Data セルは出力レイヤの NoData セルとなります
スナップで参照するレイヤ	REFERENCE_LAYER	[ラスタ]	出力レイヤの作成に使用する参照レイヤ（範囲、CRS、ピクセルの大きさ）
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	OUTPUT_NODATA_VA	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: `native:cellstackpercentile`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

セルスタック：ラスタ値のパーセントランク

ラスタのスタックのセル単位のパーセントランク値を、入力値ラスタに基づいて計算し、出力ラスタに書き出します。

各セル位置において、値ラスタの現在の値は、入力ラスタの全てのセル値を重ねてソートしたスタックのそれぞれの数値群の中で順位付けされます。スタック値の分布から外れた値は、セル値の中で順位付けできないので、 アルゴリズムは NoData を返します。

百分位数の計算には二つの方法があります:

- 線形内挿 (PERCENTRANK.INC)
- 線形内挿 (PERCENTRANK.EXC)

線形補間法は、異なるパーセンタイルにユニークな値を返します。どちらの補間法も LibreOffice や Microsoft Excel で実装されているそれぞれの方法に従っています。

出力ラスタの範囲と解像度は、参照ラスタによって決まります。参照ラスタレイヤのセルサイズと一致しない入力ラスタレイヤは、最近傍再サンプリングを使用して再サンプリングされます。「NoData を無視」パラメータが設定されていない場合、入力レイヤのいずれかに NoData 値があると、NoData セル出力になります。出力されるラスタのデータ型は常に `Float32` となります。

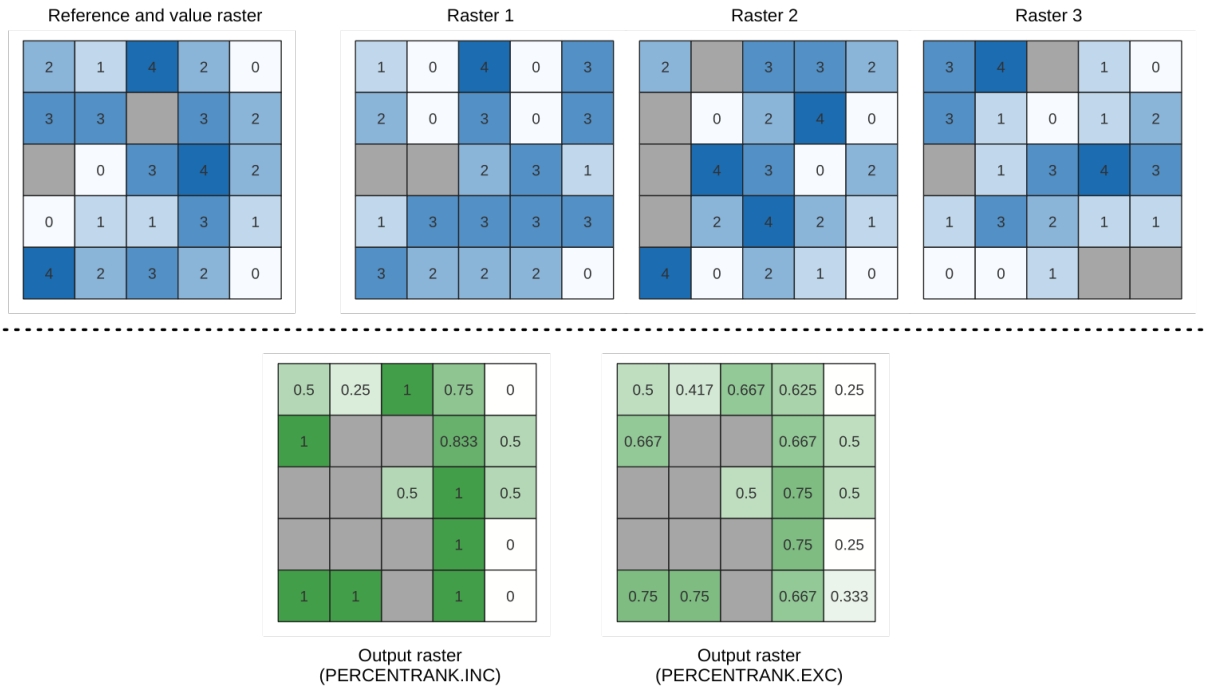


図 27.12: 値ラスタレイヤセルを順位付けします。`NoData` セル（灰色）は無視されます。

参考:

セルスタックの百分位数, **ラスタ値のパーセントランク**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	評価したいラスタレイヤ群。データラスタスタックにマルチバンドラスタを使用する場合には、このアルゴリズムは常にラスタの最初のバンドに対して解析を行う
入力ラスタ	INPUT_VALUE_RAST	[ラスタ]	重ねられたレイヤのスタックの中で値を順位付けするレイヤ
対象バンド	VALUE_RASTER_BAN	[整数] デフォルト：1	比較する「入力ラスタ」のバンド
方法	METHOD	[列挙型] デフォルト：0	百分位数の計算の方法 • 0 --- 線形内挿 (PERCENTRANK.INC) • 1 --- 線形内挿 (PERCENTRANK.EXC)
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト：True	チェックしないとき、入力レイヤの NoData セルは出力レイヤの NoData セルとなります
スナップで参照するレイヤ	REFERENCE_LAYER	[ラスタ]	出力レイヤの作成に使用する参照レイヤ（範囲、CRS、ピクセルの大きさ）
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	OUTPUT_NODATA_VA	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: `native:cellstackpercentrankfromrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

セル統計量

入力ラスタレイヤに基づいてセル毎の統計量を計算し、出力ラスタの各セルに結果の統計量を書き込みます。各セル位置において、入力ラスタのセルの値の全ての重ね合わせに対する関数値で出力値が定義されます。

デフォルトでは、任意の入力レイヤに nodata セルがあると、出力ラスタは nodata セルとなります。 *nodata* **を無視する** オプションにチェックが入っている場合は、nodata の入力値は統計量の計算において無視されます。これにより、ある位置ですべての入力レイヤが nodata のセルである場合に、結果が nodata のセルとなります。

スナップで参照するレイヤ パラメータは、出力ラスタの作成時に参照として使用する既存のラスタレイヤを指定します。出力ラスタは、このレイヤと同じ範囲、CRS、およびピクセル寸法を持ちます。

統計量計算の詳細： 参照ラスタレイヤのセルサイズと一致しない入力ラスタレイヤについては、 最近傍リサンプリング を使用して値がリサンプリングされます。出力ラスタのデータ型は、入力ラスタデータセットのうち最も複雑なデータ型になります。ただし、 平均 と 標準偏差 、 分散（Variance） は入力の浮動小数点型に応じて常に Float32 または Float64 となり、 カウント（Count） と 種類（Variety） は常に Int32 となります。

- カウント（Count） ：カウント統計は、現在のセル位置で nodata 値以外のセルの数になります。

- 中央値 ：入力レイヤのセルが偶数個の場合には、中央値はセル入力値を並べた中央の 2 つの値の算術平均で計算されます。
- 最稀値（Minority）/最頻値（Majority） ：最稀値や最頻値が一つに決まらない場合には、入力セルの値がすべて等しいのでない限り、結果は nodata となります。

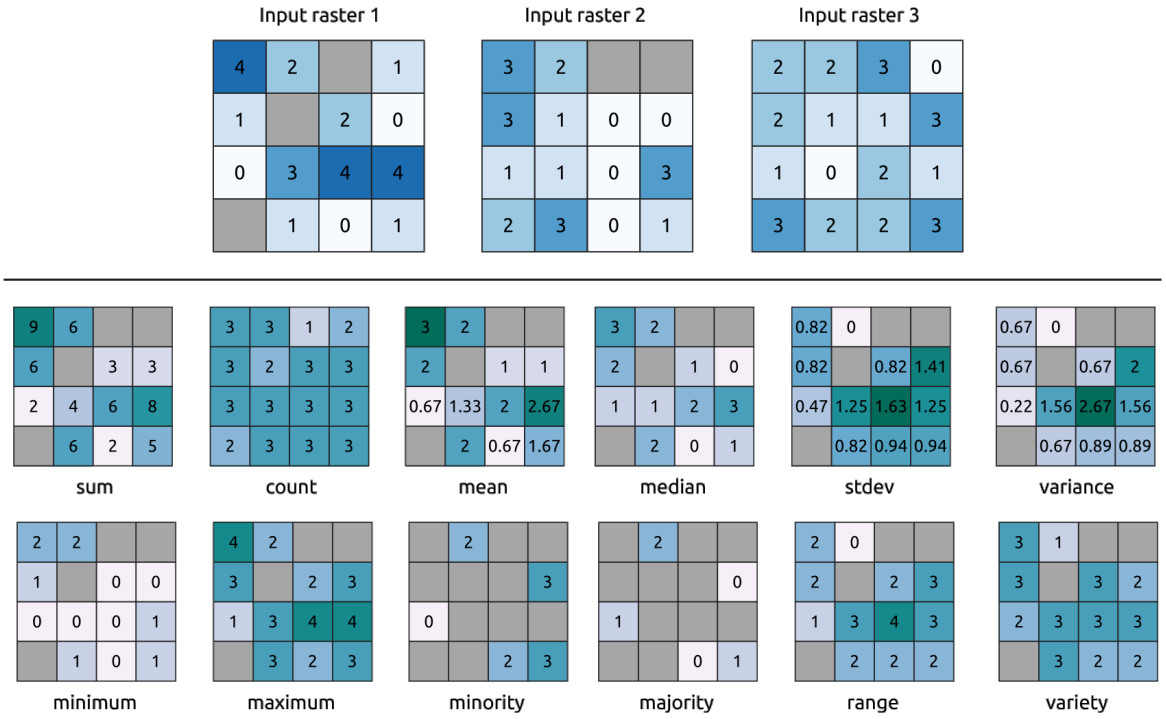


図 27.13: 全ての統計関数の例。 nodata （灰色）セルが考慮される

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	入力ラスタレイヤ
統計量	STATISTIC	[列挙型] デフォルト： 0	利用可能な統計量。オプションは次のとおり： • 0 --- 合計 • 1 --- カウント（Count） • 2 --- 平均 • 3 --- 中央値 • 4 --- 標準偏差 • 5 --- 分散（Variance） • 6 --- 最小値 • 7 --- 最大 • 8 --- 最稀値（Minority） • 9 --- 最頻値（Majority） • 10 --- 範囲（Range） • 11 --- 種類（Variety）
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト： True	nodata の存在を無視してすべてのセルの統計量を計算します。
スナップで参照するレイヤ	REF_LAYER	[ラスタ]	出力レイヤの作成に参照するレイヤ（範囲、CRS、ピクセルの大きさ等）
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する オプション	OUTPUT_NO_DATA_V	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数

Python コード

Algorithm ID: native:cellstatistics

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタスタックの値の一致頻度

入力ラスタスタックの値が値ラスタレイヤの値と一致する頻度（回数）をセル単位で評価します。出力ラスタの範囲と解像度は入力ラスタと同じで、データ型は常に Int32 です。

データラスタスタックにマルチバンドラスタを使用する場合、このアルゴリズムは常にラスタの最初のバンドに対して解析を行います。解析に他のバンドを使用したい場合には、GDAL を使用してください。出力の nodata 値は手入力で設定できます。

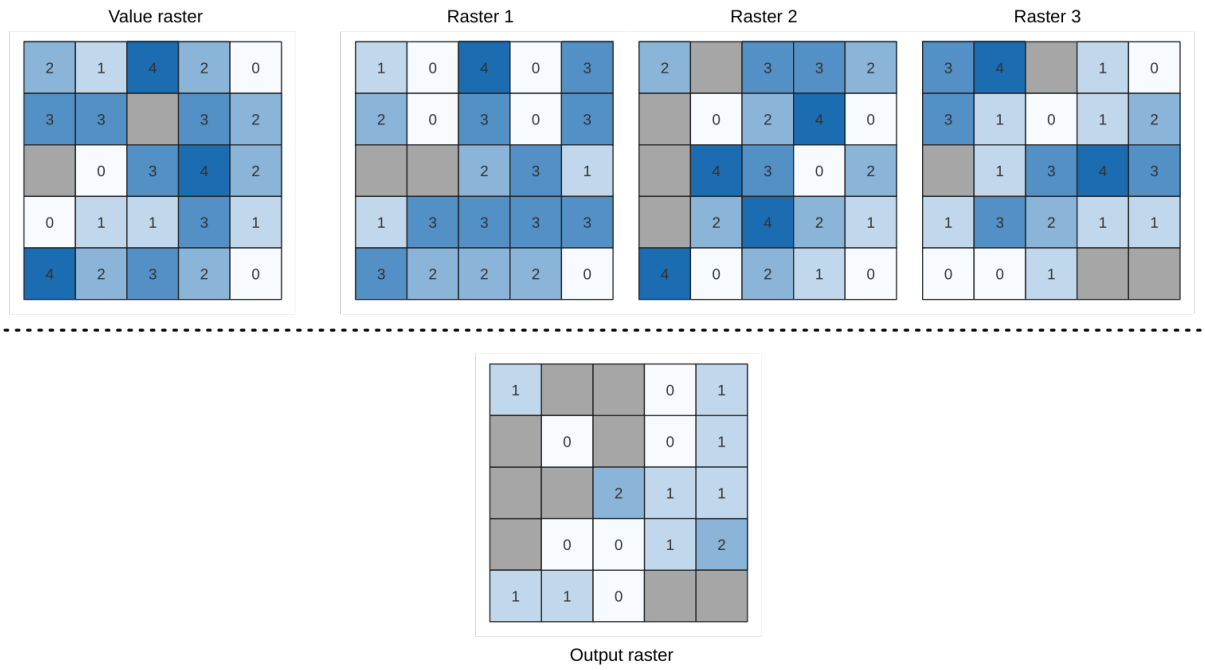


図 27.14: 出力ラスタの各セルについて、セルの値はラスタスタックのラスタの値が対応する値ラスタと同じである回数を表す。 nodata セル（灰色）が考慮される

参考:

ラスタスタックの値の超過頻度 、 **ラスタスタックの値の未満頻度**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT_VALUE_RAST	[ラスタ]	サンプリングするレイヤ群に対する参照レイヤとなる、入力値レイヤ
対象バンド	INPUT_VALUE_RAST	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	値を取得したいバンドを選択します
入力ラスタ（複数）	INPUT_RASTERS	[ラスタ] [リスト]	評価したいラスタレイヤ群。データラスタスタックにマルチバンドラスタを使用する場合には、このアルゴリズムは常にラスタの最初のバンドに対して解析を行う

次のページに続く

表 27.49 – 前のページからの続き

ラベル	名前	データ型	説明
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト：False	チェックを入れない場合、値ラスタやデータレイヤスタックに nodata セルが一つでもあれば、出力ラスタは nodata セルとなります。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力するオプション	OUTPUT_NO_DATA_V	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
同じ出現頻度のセル数	FOUND_LOCATIONS_	[数値]	
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[数値]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
有効セルの場所の平均頻度	MEAN_FREQUENCY_P	[数値]	
値の頻度	OCCURRENCE_COUNT	[数値]	
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数

Python コード

Algorithm ID: `native:equaltofrequency`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのファジー化（gaussian membership）

ガウシアンメンバーシップ関数を使用して入力ラスタの各ピクセルにメンバーシップ値を割り当てることにより、入力ラスタをファジー化されたラスタに変換します。メンバーシップ値は 0 から 1 の範囲の値です。ファジー化されたラスタにおいて、値 0 はメンバーではないこと、値 1 は完全にメンバーであることを意味しています。ガウシアンメンバーシップ関数は、 $\mu(x) = e^{-f1*(x-f2)^2}$ で定義されます。ここで、*f1* は拡がり、*f2* は中点を表します。

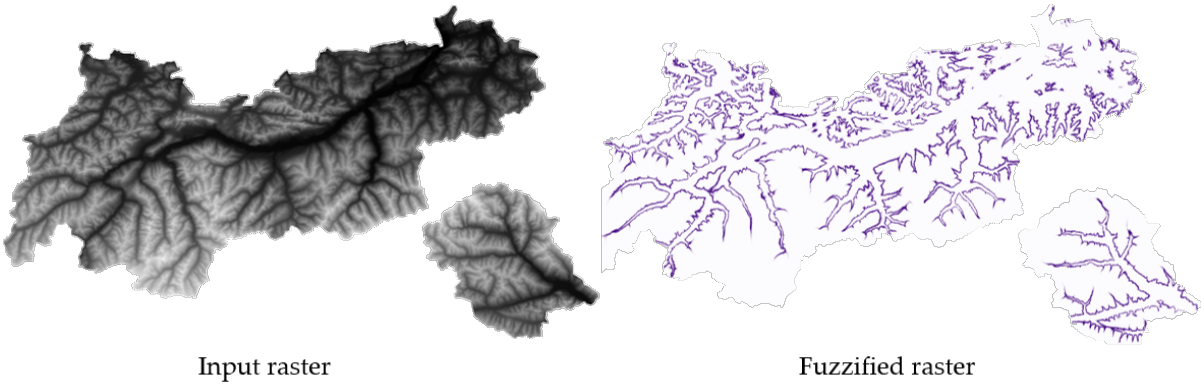


図 27.15: ラスタのファジー化の例。入力ラスタソース：Land Tirol - data.tirol.gv.at

参考:

ラスタのファジー化（*large membership*） 、 **ラスタのファジー化（*linear membership*）** 、 **ラスタのファジー化（*near membership*）** 、 **ラスタのファジー化（*power membership*）** 、 **ラスタのファジー化（*small membership*）**

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ファジー化したいバンドを選択してください。
Function mid-point	FUZZYMIDPOINT	[数値] デフォルト： 10	ガウス関数の中点
Function spread	FUZZYSPREAD	[数値] デフォルト： 0.01	ガウス関数の拡がり
出力ラスタ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[入力レイヤと同じ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: `native:fuzzifyrastergaussianmembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのファジー化（large membership）

large メンバーシップ関数を使用して入力ラスタの各ピクセルにメンバーシップ値を割り当てることにより、入力ラスタをファジー化されたラスタに変換します。メンバーシップ値は 0 から 1 の範囲の値です。ファジー化されたラスタにおいて、値 0 はメンバーではないこと、値 1 は完全にメンバーであることを意味しています。large メンバーシップ関数は、$\mu(x) = \frac{1}{1 + \left(\frac{x}{f2}\right)^{-f1}}$ で定義されます。ここで、*f1* は拡がり、*f2* は中点を表します。

参考:

ラスタのファジー化（*gaussian membership*）、**ラスタのファジー化（*linear membership*）**、**ラスタのファジー化（*near membership*）**、**ラスタのファジー化（*power membership*）**、**ラスタのファジー化（*small membership*）**

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	`INPUT`	[ラスタ]	入力ラスタレイヤ
バンド番号	`BAND`	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ファジー化したいバンドを選択してください。
Function mid-point	`FUZZYMIDPOINT`	[数値] デフォルト： 50	large 関数の中点
Function spread	`FUZZYSPREAD`	[数値] デフォルト： 5	large 関数の拡がり
出力ラスタ	`OUTPUT`	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[入力レイヤと同じ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: native:fuzzifyrasterlargemembership

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのファジー化（linear membership）

線形メンバーシップ関数を使用して入力ラスタの各ピクセルにメンバーシップ値を割り当てることにより、入力ラスタをファジー化されたラスタに変換します。メンバーシップ値は 0 から 1 の範囲の値です。ファジー化されたラスタにおいて、値 0 はメンバーではないこと、値 1 は完全にメンバーであることを意味しています。線形メンバーシップ関数は、$\mu(X)\begin{cases}0 & x \leq a \\ \frac{x-a}{b-a} & a < x < b \\ 1 & x \geq b\end{cases}$ で定義されます。ここで、 a は下限、 b は上限を表します。この式は、上下限の間でピクセル値を線形変換してメンバーシップ値を割り当てます。下限よりも小さなピクセル値はメンバーシップ値 0、上限よりも大きなピクセル値はメンバーシップ値 1 となります。

参考:

ラスタのファジー化（*gaussian membership*） 、 **ラスタのファジー化（*large membership*）** 、 **ラスタのファジー化（*near membership*）** 、 **ラスタのファジー化（*power membership*）** 、 **ラスタのファジー化（*small membership*）**

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ファジー化したいバンドを選択してください。
Low fuzzy membership bound	FUZZYLOWBOUND	[数値] デフォルト： 0	線形関数の下限
High fuzzy membership bound	FUZZYHIGHBOUND	[数値] デフォルト： 1	線形関数の上限
出力ラスタ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[入力レイヤと同じ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: native:fuzzifyrasterlinearmembership

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのファジー化（near membership）

near メンバーシップ関数を使用して入力ラスタの各ピクセルにメンバーシップ値を割り当てることにより、入力ラスタをファジー化されたラスタに変換します。メンバーシップ値は 0 から 1 の範囲の値です。ファジー化されたラスタにおいて、値 0 はメンバーではないこと、値 1 は完全にメンバーであることを意味しています。near メンバーシップ関数は、$\mu(x) = \frac{1}{1 + f1 * (x - f2)^2}$ で定義されます。ここで、*f1* は拡がり、*f2* は中点を表します。

参考:

ラスタのファジー化（*gaussian membership*）、**ラスタのファジー化（*large membership*）**、**ラスタのファジー化（*linear membership*）**、**ラスタのファジー化（*power membership*）**、**ラスタのファジー化（*small membership*）**

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	`INPUT`	[ラスタ]	入力ラスタレイヤ
バンド番号	`BAND`	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ファジー化したいバンドを選択してください。
Function mid-point	`FUZZYMIDPOINT`	[数値] デフォルト： 50	near 関数の中点
Function spread	`FUZZYSPREAD`	[数値] デフォルト： 0.01	near 関数の拡がり
出力ラスタ	`OUTPUT`	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[入力レイヤと同じ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: native:fuzzifyrasternearmembership

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのファジー化（power membership）

べき乗メンバーシップ関数を使用して入力ラスタの各ピクセルにメンバーシップ値を割り当てることにより、入力ラスタをファジー化されたラスタに変換します。メンバーシップ値は 0 から 1 の範囲の値です。ファジー化されたラスタにおいて、値 0 はメンバーではないこと、値 1 は完全にメンバーであることを意味しています。べき乗メンバーシップ関数は、

$$\mu(x)\begin{cases}0 & x \le a\\ \left(\frac{x-a}{b-a}\right)^{f1} & a < x < b\\ 1 & x \ge b\end{cases}$$

で定義されます。ここで、 *a* は下限、 *b* は上限、 *f1* は指数を表します。この式は、上下限の間でピクセル値をべき乗変換してメンバーシップ値を割り当てます。下限よりも小さなピクセル値はメンバーシップ値 0、上限よりも大きなピクセル値はメンバーシップ値 1 となります。

参考:

ラスタのファジー化（*gaussian membership*） 、 **ラスタのファジー化（*large membership*）** 、 **ラスタのファジー化（*linear membership*）** 、 **ラスタのファジー化（*near membership*）** 、 **ラスタのファジー化（*small membership*）**

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ファジー化したいバンドを選択してください。
Low fuzzy membership bound	FUZZYLOWBOUND	[数値] デフォルト： 0	べき乗関数の下限
High fuzzy membership bound	FUZZYHIGHBOUND	[数値] デフォルト： 1	べき乗関数の上限
High fuzzy membership bound	FUZZYEXPONENT	[数値] デフォルト： 2	べき乗関数の指数
出力ラスタ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[入力レイヤと同じ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: native:fuzzifyrasterpowermembership

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのファジー化（small membership）

small メンバーシップ関数を使用して入力ラスタの各ピクセルにメンバーシップ値を割り当てることにより、入力ラスタをファジー化されたラスタに変換します。メンバーシップ値は 0 から 1 の範囲の値です。ファジー化されたラスタにおいて、値 0 はメンバーではないこと、値 1 は完全にメンバーであることを意味しています。small メンバーシップ関数は、$\mu(x) = \frac{1}{1+\left(\frac{x}{f2}\right)^{f1}}$ で定義されます。ここで、*f1* は拡がり、*f2* は中点を表します。

参考:

ラスタのファジー化（*gaussian membership*） 、 **ラスタのファジー化（*large membership*）** 、 **ラスタのファジー化（*linear membership*）** 、 **ラスタのファジー化（*near membership*）** 、 **ラスタのファジー化（*power membership*）**

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	`INPUT`	[ラスタ]	入力ラスタレイヤ
バンド番号	`BAND`	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ファジー化したいバンドを選択してください。
Function mid-point	`FUZZYMIDPOINT`	[数値] デフォルト： 50	small 関数の中点
Function spread	`FUZZYSPREAD`	[数値] デフォルト： 5	small 関数の拡がり
出力ラスタ	`OUTPUT`	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[入力レイヤと同じ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: `native:fuzzifyrastersmallmembership`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタスタックの値の超過頻度

入力ラスタスタックの値が値ラスタレイヤの値よりも大きい頻度（回数）をセル単位で評価します。出力ラスタの範囲と解像度は入力ラスタと同じで、データ型は常に `Int32` です。

データラスタスタックにマルチバンドラスタを使用する場合、このアルゴリズムは常にラスタの最初のバンドに対して解析を行います。解析に他のバンドを使用したい場合には、GDAL を使用してください。出力の nodata 値は手入力で設定できます。

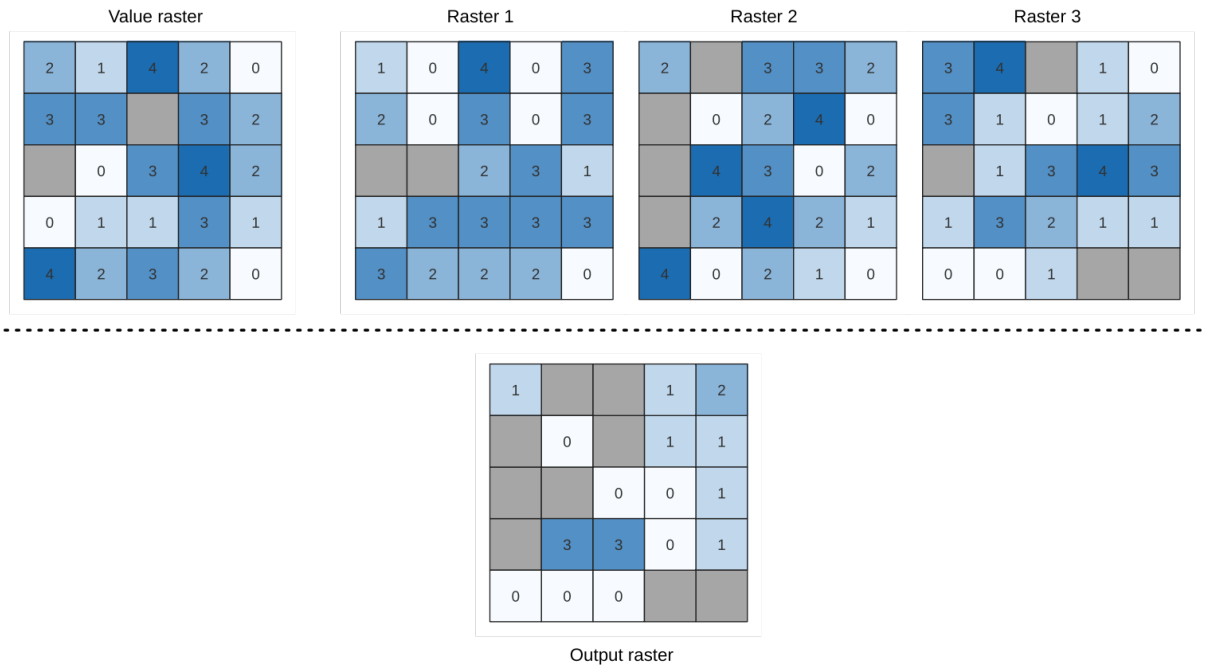


図 27.16: 出力ラスタの各セルについて、セルの値はラスタスタックのラスタの値が対応する値ラスタよりも大きい回数を表す。 nodata セル（灰色）が考慮される

参考:

ラスタスタックの値の一致頻度 、 **ラスタスタックの値の未満頻度**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT_VALUE_RAST	[ラスタ]	サンプリングするレイヤ群に対する参照レイヤとなる、入力値レイヤ
対象バンド	INPUT_VALUE_RAST	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	値を取得したいバンドを選択します
入力ラスタ（複数）	INPUT_RASTERS	[ラスタ] [リスト]	評価したいラスタレイヤ群。データラスタスタックにマルチバンドラスタを使用する場合には、このアルゴリズムは常にラスタの最初のバンドに対して解析を行う

次のページに続く

表 27.52 – 前のページからの続き

ラベル	名前	データ型	説明
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト：False	チェックを入れない場合、値ラスタやデータレイヤスタックに nodata セルが一つでもあれば、出力ラスタは nodata セルとなります。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力するオプション	OUTPUT_NO_DATA_V	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
同じ出現頻度のセル数	FOUND_LOCATIONS_	[数値]	
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[数値]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
有効セルの場所の平均頻度	MEAN_FREQUENCY_P	[数値]	
値の頻度	OCCURRENCE_COUNT	[数値]	
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数

Python コード

Algorithm ID: `native:greaterthanfrequency`

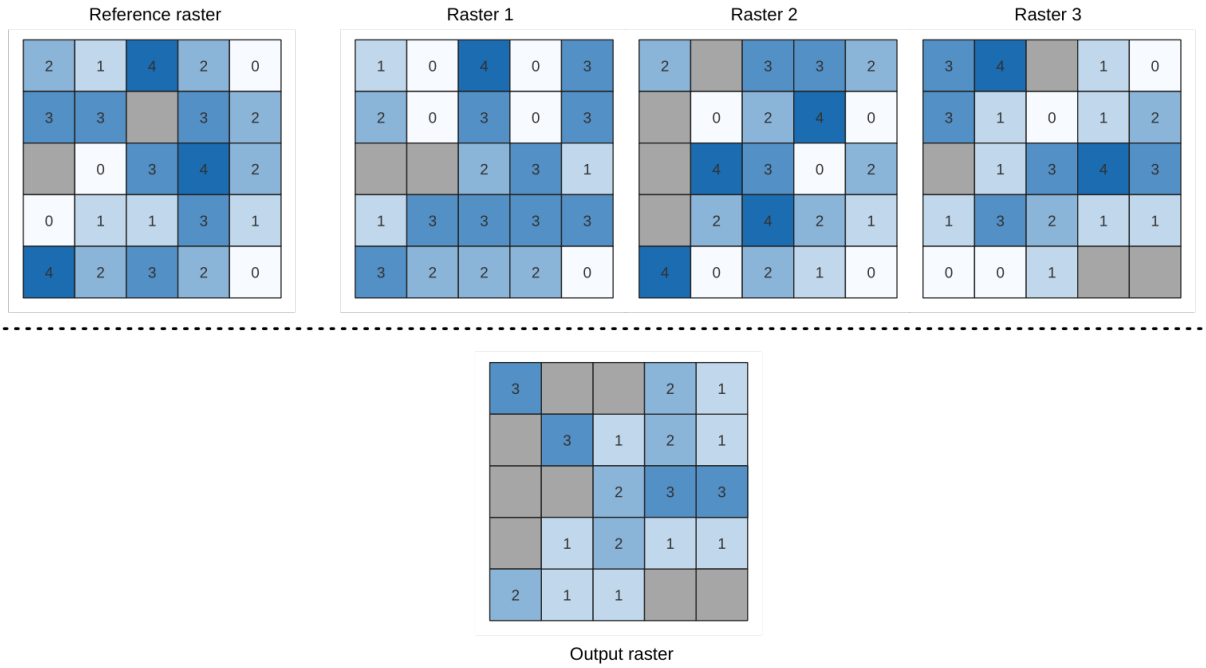
```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタスタックの最大値の位置

入力ラスタスタック内で最大値をとるラスタの位置をセル単位で評価します。位置のカウントは 1 から始まり、入力ラスタの総数までの値をとります。このアルゴリズムでは、入力ラスタの順序が関係します。複数のラスタが最大値となる場合、最初のラスタが位置の値として使用されます。

データラスタスタックにマルチバンドラスタを使用する場合、このアルゴリズムは常にラスタの最初のバンドに対して解析を行います。解析に他のバンドを使用したい場合には、GDAL を使用してください。ラスタレイヤスタックのセルに nodata が一つでもあれば、 "nodata を無視する" パラメータにチェックを入れていない限りは、出力ラスタは nodata セルとなります。出力の nodata 値は手入力で設定できます。出力ラスタの範囲と解像度は参照ラスタレイヤで定義され、データ型は常に `Int32` です。



参考:

ラスタスタックの最小値の位置

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ラスタ（複数）	INPUT_RASTERS	[ラスタ] [リスト]	比較を行うラスタレイヤのリスト
スナップで参照するレイヤ	REFERENCE_LAYER	[ラスタ]	出力レイヤの作成に使用する参照レイヤ（範囲、CRS、ピクセルの大きさ）
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト：False	チェックを入れない場合、データレイヤスタックに nodata セルが一つでもあれば、出力ラスタは nodata セルとなります。
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	結果の出力レイヤを指定します。つぎのいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	OUTPUT_NODATA_VA	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

Algorithm ID: `native:highestpositioninrasterstack`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタスタックの値の未満頻度

入力ラスタスタックの値が値ラスタレイヤの値よりも小さい頻度（回数）をセル単位で評価します。出力ラスタの範囲と解像度は入力ラスタと同じで、データ型は常に `Int32` です。

データラスタスタックにマルチバンドラスタを使用する場合、このアルゴリズムは常にラスタの最初のバンドに対して解析を行います。解析に他のバンドを使用したい場合には、GDAL を使用してください。出力の nodata 値は手入力で設定できます。

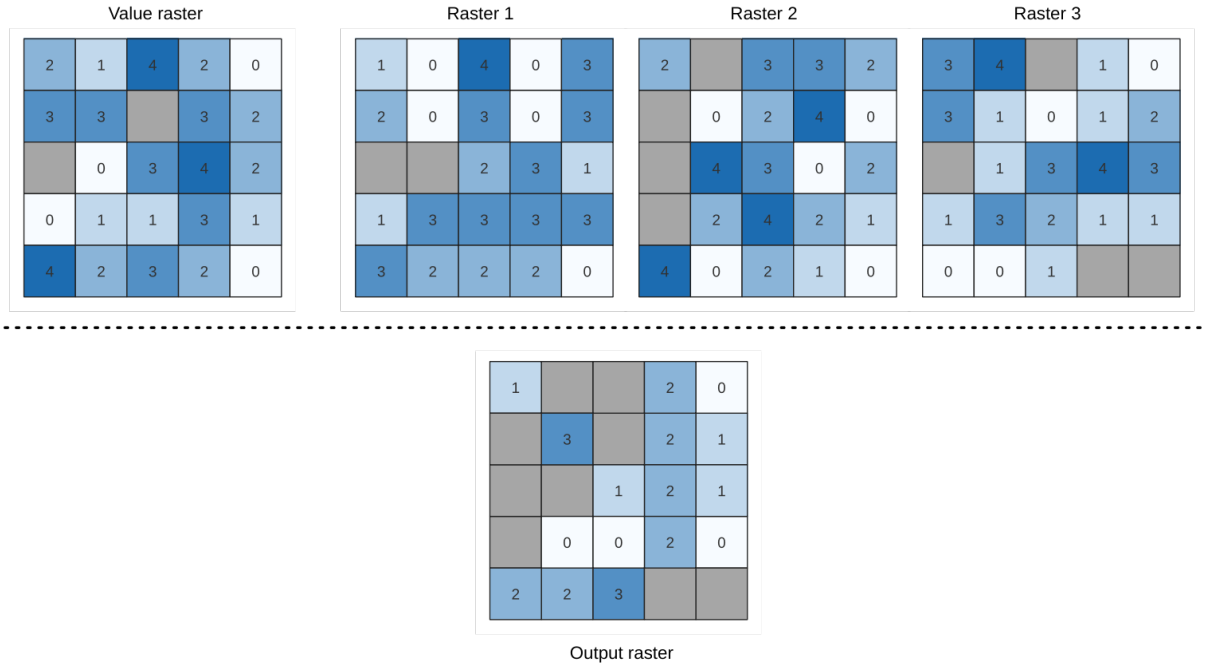


図 27.17: 出力ラスタの各セルについて、セルの値はラスタスタックのラスタの値が対応する値ラスタよりも小さい回数を表す。 `nodata` セル（灰色）が考慮される

参考:

ラスタスタックの値の一致頻度 、 **ラスタスタックの値の超過頻度**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT_VALUE_RAST	[ラスタ]	サンプリングするレイヤ群に対する参照レイヤとなる、入力値レイヤ
対象バンド	INPUT_VALUE_RAST	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	値を取得したいバンドを選択します
入力ラスタ（複数）	INPUT_RASTERS	[ラスタ] [リスト]	評価したいラスタレイヤ群。データラスタスタックにマルチバンドラスタを使用する場合には、このアルゴリズムは常にラスタの最初のバンドに対して解析を行う
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト：False	チェックを入れない場合、値ラスタやデータレイヤスタックに nodata セルが一つでもあれば、出力ラスタは nodata セルとなります。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する オプション	OUTPUT_NO_DATA_V	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
同じ出現頻度のセル数	FOUND_LOCATIONS_	[数値]	
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[数値]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
有効セルの場所の平均頻度	MEAN_FREQUENCY_P	[数値]	
値の頻度	OCCURRENCE_COUNT	[数値]	
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数

Python コード

Algorithm ID: `native:lessthanfrequency`

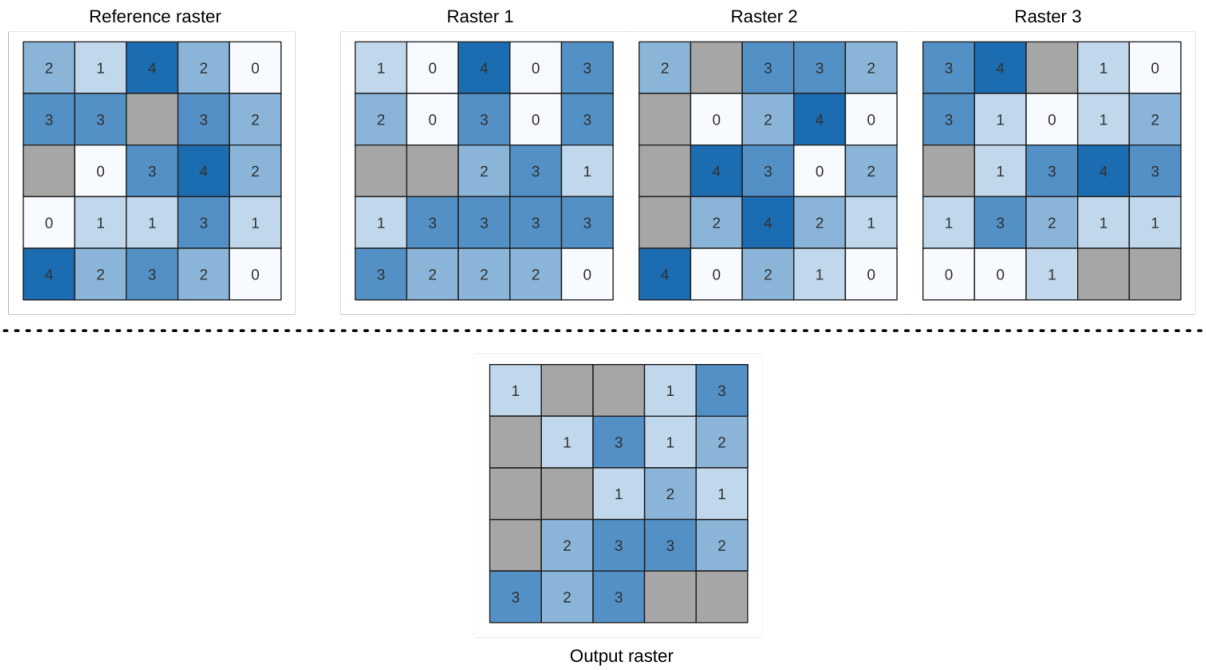
```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタスタックの最小値の位置

入力ラスタスタック内で最小値をとるラスタの位置をセル単位で評価します。位置のカウントは 1 から始まり、入力ラスタの総数までの値をとります。このアルゴリズムでは、入力ラスタの順序が関係します。複数のラスタが最小値となる場合、最初のラスタが位置の値として使用されます。

データラスタスタックにマルチバンドラスタを使用する場合、このアルゴリズムは常にラスタの最初のバンドに対して解析を行います。解析に他のバンドを使用したい場合には、GDAL を使用してください。ラスタレイヤスタックのセルに nodata が一つでもあれば、 "nodata を無視する" パラメータにチェックを入れていない限りは、出力ラスタは nodata セルとなります。出力の nodata 値は手入力で設定できます。出力ラスタの範囲と解像度は参照ラスタレイヤで定義され、データ型は常に `Int32` です。



参考:

ラスタスタックの最大値の位置

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ラスタ（複数）	INPUT_RASTERS	[ラスタ] [リスト]	比較を行うラスタレイヤのリスト
スナップで参照するレイヤ	REFERENCE_LAYER	[ラスタ]	出力レイヤの作成に使用する参照レイヤ（範囲、CRS、ピクセルの大きさ）
nodata を無視する	IGNORE_NODATA	[ブール値] デフォルト：False	チェックを入れない場合、データレイヤスタックに nodata セルが一つでもあれば、出力ラスタは nodata セルとなります。
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	結果の出力レイヤを指定します。つぎのいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	OUTPUT_NODATA_VA	[数値] デフォルト: -9999.0	出力レイヤの nodata に使用する値

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数

Python コード

アルゴリズム ID: `native:lowestpositioninrasterstack`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

複数ラスタの AND 論理値

一連の入力ラスタの論理 AND を計算します。あるピクセルについて、すべての入力ラスタが非ゼロ値をもつ場合、そのピクセルは出力ラスタで 1 になります。入力ラスタのいずれかに 0 値がある場合、そのピクセルは出力ラスタで 0 になります。

参照レイヤパラメータには、出力ラスタの作成時に参照として使用する既存のラスタレイヤを指定します。出力ラスタは、このレイヤと同じ範囲、CRS、およびピクセル寸法を持ちます。

デフォルトでは、任意の入力レイヤに nodata ピクセルがあると、出力ラスタは nodata ピクセルとなります。 *nodata* **を** *false* **とみなす** オプションにチェックが入っている場合には、nodata の入力値は入力値 0 と同じものとして扱われます。

参考:

複数ラスタの *OR* 論理値

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	入力ラスタレイヤのリスト
スナップで参照するレイヤ	REF_LAYER	[ラスタ]	出力レイヤの作成に参照するレイヤ（範囲、CRS、ピクセルの大きさ等）
nodata を false とみなす	NODATA_AS_FALSE	[ブール値] デフォルト：False	演算を実行する際に、入力ファイルの nodata 値を 0 として扱います
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	結果の出力レイヤを指定します。つぎのいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	NO_DATA	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値

次のページに続く

表 27.61 – 前のページからの続き

ラベル	名前	データ型	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタのデータ型。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 --- Int32 • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	データ型	説明
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
nodata ピクセルの数	NODATA_PIXEL_COU	[整数]	出力ラスタレイヤの nodata ピクセルの数
True のピクセル数	TRUE_PIXEL_COUNT	[整数]	出力ラスタレイヤの True ピクセル（value=1）の数
False のピクセル数	FALSE_PIXEL_COUN	[整数]	出力ラスタレイヤの False ピクセル（value=0）の数
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ

Python コード

Algorithm ID: `native:rasterbooleanand`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

複数ラスタの OR 論理値

一連の入力ラスタの論理 OR を計算します。あるピクセルについて、すべての入力ラスタがゼロ値をもつ場合、そのピクセルは出力ラスタで 0 になります。入力ラスタのいずれかに 1 値がある場合、そのピクセルは出力ラスタで 1 になります。

参照レイヤパラメータには、出力ラスタの作成時に参照として使用する既存のラスタレイヤを指定します。出力ラスタは、このレイヤと同じ範囲、CRS、およびピクセル寸法を持ちます。

デフォルトでは、任意の入力レイヤに nodata ピクセルがあると、出力ラスタは nodata ピクセルとなります。 *nodata* **を** *false* **とみなす** オプションにチェックが入っている場合には、nodata の入力値は入力値 0 と同じものとして扱われます。

参考:

複数ラスタの *AND* **論理値**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	入力ラスタレイヤのリスト
スナップで参照するレイヤ	REF_LAYER	[ラスタ]	出力レイヤの作成に参照するレイヤ（範囲、CRS、ピクセルの大きさ等）
nodata を false とみなす	NODATA_AS_FALSE	[ブール値] デフォルト：False	演算を実行する際に、入力ファイルの nodata 値を 0 として扱います
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	結果の出力レイヤを指定します。つぎのいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	NO_DATA	[数値] デフォルト：-9999.0	出力レイヤの nodata に使用する値
出力のデータ型	DATA_TYPE	[列挙型] デフォルト：5	出力ラスタのデータ型。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 --- Int32 • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	データ型	説明
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
CRS 権限識別子	CRS_AUTHID	[crs]	出力ラスタレイヤの座標参照系
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
nodata ピクセルの数	NODATA_PIXEL_COU	[整数]	出力ラスタレイヤの nodata ピクセルの数

次のページに続く

表 27.65 – 前のページからの続き

ラベル	名前	データ型	説明
True のピクセル数	TRUE_PIXEL_COUNT	[整数]	出力ラスタレイヤの True ピクセル（value=1）の数
False のピクセル数	FALSE_PIXEL_COUN	[整数]	出力ラスタレイヤの False ピクセル（value=0）の数
出力レイヤ	OUTPUT	[ラスタ]	結果が含まれる出力ラスタレイヤ

Python コード

Algorithm ID: native:rasterbooleanor

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタ計算機

ラスタレイヤを使用して代数演算を実行します。

結果のレイヤは、式に従って計算された値になります。式には、数値、演算子、現在のプロジェクト内の任意のレイヤへの参照を含めることができます。

注釈: **バッチ処理インターフェイス** または *QGIS Python* **コンソール** から計算機を使用するときは、使用するファイルを指定する必要があります。対応するレイヤは、ファイルのベース名を使用して参照されます（フルパスなし）。例えば、 path/to/my/rasterfile.tif にあるレイヤーを使用している場合、そのレイヤーの最初のバンドは rasterfile.tif@1 として参照されます。

参考:

ラスタ計算機

パラメータ

ラベル	名前	データ型	説明
レイヤ	GUI 版のみ		凡例に読み込まれているすべてのラスタレイヤのリストが表示されます。これを使って式ボックスに入力できます（ダブルクリックで追加）。ラスタレイヤは、 layer_name@band_number のように、その名前とバンド番号で参照されます。例えば、DEM という名前のレイヤの最初のバンドは、 DEM@1 という名前で参照されます。
演算子	GUI 版のみ		電卓のようなボタンがあり、これを使って式ボックスに入力できます。
式	EXPRESSION	[文字列]	出力ラスタレイヤの計算に使用する式。用意されている演算子ボタンを使用して、このボックスに式を直接入力できます。
定義済みの式	GUI 版のみ		定義済みの NDVI 式を使用したり、新たに式を定義して計算できます。 **追加**... ボタンを押すと、定義済みの式を読み込みます（さらにパラメータの設定ができます）。 **保存**... ボタンを押すと、新しい式を定義できます。
Reference layer(s) (used for automated extent, cellsize, and CRS) オプション	LAYERS	[ラスタ] [リスト]	範囲やセルサイズ、CRS を取得するために使用するレイヤ。このボックスでレイヤを選択すると、その他全てのパラメータを手入力せずに済みます。ラスタレイヤは、 layer_name@band_number のように、その名前とバンド番号で参照されます。例えば、DEM という名前のレイヤの最初のバンドは、 DEM@1 という名前で参照されます。
Cell size (use 0 or empty to set it automatically) オプション	CELLSIZE	[数値]	出力ラスタレイヤのセルサイズ。セルサイズが指定されていない場合は、選択した参照レイヤの最小セルサイズが使用されます。セルサイズは、X 軸と Y 軸の両方で同じ長さです。

次のページに続く

表 27.66 – 前のページからの続き

ラベル	名前	データ型	説明
出力領域 オプション	EXTENT	[範囲]	出力ラスタレイヤの空間範囲を指定します。範囲が指定されていない場合は、選択された参照レイヤを全てカバーする最小範囲が使用されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
Output CRS オプション	CRS	[crs]	出力ラスタレイヤの CRS。出力 CRS が指定されていない場合は、最初の参照レイヤの CRS が使用されます。
出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力	OUTPUT	[ラスタ]	計算した値の出力ラスタファイル。

Python コード

Algorithm ID: `qgis:rastercalculator`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタレイヤのプロパティ

指定されたラスタレイヤの基本プロパティ（範囲、ピクセル単位の大きさ、（地図単位での）ピクセルの寸法、バンド数、データなし値）を返します。

このアルゴリズムは、モデル内の他のアルゴリズムへの入力値として使用するために、これらの有用なプロパティを抽出する手段として使用することを意図しています。例えば、存在するラスタのピクセルの大きさを GDAL ラスタアルゴリズムに渡すことができます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号 オプション	BAND	[ラスタのバンド] デフォルト：未設定	特定のバンドのプロパティも返すかどうか。バンドが指定された場合、選択されたバンドの noData 値も返されます。

出力

ラベル	名前	データ型	説明
ラスタにあるバンドの数	BAND_COUNT	[数値]	ラスタにあるバンドの数
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	CRS によるラスタレイヤの範囲
バンドは設定された NoData 値を持っている	HAS_NODATA_VALUE	[ブール値]	ラスタレイヤが選択されたバンドに NODATA ピクセル用に設定された値を持っているかどうかを示す
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	ラスタレイヤにあるカラムの数
バンドの NoData 値	NODATA_VALUE	[数値]	（設定されていれば、）選択したバンドにおける、NoData ピクセルの値
マップの単位によるピクセルの大きさ（高さ）	PIXEL_HEIGHT	[整数]	ピクセルのマップの単位による垂直方向の大きさ
マップの単位によるピクセルの大きさ（幅）	PIXEL_WIDTH	[整数]	ピクセルのマップの単位による水平方向の大きさ
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	ラスタレイヤにある行の数
x 座標の最大	X_MAX	[数値]	
x 座標の最小	X_MIN	[数値]	
y 座標の最大	Y_MAX	[数値]	
y 座標の最小	Y_MIN	[数値]	

Python コード

Algorithm ID: native:rasterlayerproperties

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタレイヤの統計量

ラスタレイヤの指定したバンドの値に関する基本統計量を計算します。結果は **プロセシング　結果ビューア** メニューに読み込まれます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、統計量を計算したいバンドを選択してください。
統計量の出力	OUTPUT_HTML_FILE	[html] デフォルト：[一時ファイルに保存]	出力ファイルの指定： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
最大値	MAX	[数値]	
平均値	MEAN	[数値]	
最小値	MIN	[数値]	
統計量の出力	OUTPUT_HTML_FILE	[html]	出力ファイルには以下の情報が含まれます： • 分析済みファイル : ラスタレイヤのパス • 最小値 : ラスタの最小値 • 最大値 : ラスタの最大値 • 範囲 : 最大値と最小値の差 • 合計 : 値の総和 • 平均値 : 値の平均 • 標準偏差 : 値の標準偏差 • 平方和 : 各値と全体の平均値との差の二乗の合計
範囲	RANGE	[数値]	
標準偏差	STD_DEV	[数値]	
合計	SUM	[数値]	

次のページに続く

表 27.67 – 前のページからの続き

ラベル	名前	データ型	説明
平方和	SUM_OF_SQUARES	[数値]	

Python コード

Algorithm ID: native:rasterlayerstatistics

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタレイヤのユニーク値

指定したラスタレイヤの値ごとの個数と面積を返します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、統計量を計算したいバンドを選択してください。
ユニーク値の集計出力	OUTPUT_HTML_FILE	[ファイル] デフォルト：[一時ファイルに保存]	出力ファイルの指定： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

次のページに続く

表 27.68 – 前のページからの続き

ラベル	名前	データ型	説明
ユニーク値の表	OUTPUT_TABLE	[テーブル] デフォルト：[出力をスキップ]	ユニーク値のテーブルの指定： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
nodata ピクセルの数	NODATA_PIXEL_COU	[数値]	出力ラスタレイヤの NODATA ピクセルの数
総ピクセル数	TOTAL_PIXEL_COUN	[整数]	出力ラスタレイヤのピクセル数
ユニーク値の集計出力	OUTPUT_HTML_FILE	[html]	出力 HTML ファイルには以下の情報が含まれます： • 分析ファイル : ラスタレイヤのパス • 範囲 : 範囲の x 最小値、y 最小値、x 最大値、y 最大値の座標 • 投影法 : レイヤの投影法 • 幅（ピクセル単位）: 列の数と、ピクセルの幅サイズ • 高さ（ピクセル単位）: 行の数と、ピクセルの高さサイズ • 総ピクセル数 : すべてのピクセルの個数 • nodata ピクセルの数 : nodata 値を持つピクセルの個数

次のページに続く

表 27.69 – 前のページからの続き

ラベル	名前	データ型	説明
ユニーク値の表	OUTPUT_TABLE	[テーブル]	テーブルには 3 つのカラムがあります： • *value*: ピクセルの値 • *count*: その値のピクセル数 • m^2: その値のピクセルの総面積（平方メートル単位）
幅（ピクセル単位）	WIDTH_IN_PIXELS	[整数]	出力ラスタレイヤの列数

Python コード

Algorithm ID: native:rasterlayeruniquevaluesreport

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ゾーン統計量（ラスタ）

別のラスタレイヤで定義されたゾーンで分類して、ラスタレイヤの値の統計量を計算します。

参考:

ゾーン統計量（ベクタ）

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、統計量を計算したいバンドを選択してください。

次のページに続く

表 27.70 – 前のページからの続き

ラベル	名前	データ型	説明
ゾーン統計の対象レイヤ	ZONES	[ラスタ]	ゾーンを定義するラスタレイヤ。ゾーンとは、同じピクセル値を持つ連続したピクセルです。
対象バンド	ZONES_BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、ゾーンを定義するバンドを選択してください。
統計量の出力	OUTPUT_TABLE	[テーブル] デフォルト：[一時レイヤを作成]	出力レポートを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
スナップで参照するレイヤ オプション	REF_LAYER	[列挙型] デフォルト：0	出力レイヤのゾーンの判定時に参照点として用いる重心の計算に使用するラスタレイヤ。次のいずれかです： • 0 --- 入力レイヤ：ゾーンは、ソースラスタレイヤから各ピクセルの重心でゾーンラスタレイヤの値をサンプリングすることによって決定されます • 1 --- ゾーンレイヤ：入力ラスタレイヤは、ゾーンラスタレイヤの各ピクセルの重心でサンプリングされます

出力

ラベル	名前	データ型	説明
CRS 権限識別子	CRS_AUTHID	[文字列]	出力ラスタレイヤの座標参照系
領域	EXTENT	[文字列]	出力ラスタレイヤの空間範囲
高さ（ピクセル単位）	HEIGHT_IN_PIXELS	[整数]	出力ラスタレイヤの行数
nodata ピクセルの数	NODATA_PIXEL_COU	[数値]	出力ラスタレイヤの NODATA ピクセルの数
統計量の出力	OUTPUT_TABLE	[テーブル]	出力レイヤには **各ゾーンについての** 以下の情報が含まれます： • m2: ゾーンの面積（単位はラスタの長さ単位の二乗） • sum: ゾーン内のピクセル値の総和 • count: ゾーン内のピクセルの個数 • min: ゾーン内のピクセル値の最小値 • max: ゾーン内のピクセル値の最大値 • mean: ゾーン内のピクセル値の平均値
総ピクセル数	TOTAL_PIXEL_COUN	[数値]	出力ラスタレイヤのピクセル数
幅（ピクセル単位）	WIDTH_IN_PIXELS	[数値]	出力ラスタレイヤの列数

Python コード

Algorithm ID: `native:rasterlayerzonalstats`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのサーフェス体積

指定したベースレベルを基準とした、ラスタサーフェスの下の体積を計算します。主に数値標高モデル（DEM）で便利なアルゴリズムです。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ラスタ]	サーフェスを表す入力ラスタ
バンド番号	BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、サーフェスを定義するバンドを選択してください。
基準値（ベースレベル）	LEVEL	[数値] デフォルト： 0.0	ベース、あるいは基準値を定義します。この基準値は、方法 パラメータ（以下参照）に応じた体積計算に用いられます。
方法	METHOD	[列挙型] デフォルト： 0	ラスタピクセル値と 基準値 の差による体積計算の方法を定義します。オプションは次のとおり： • 0 --- 基準値を超える場合だけ：基準値を上回るピクセルのみが体積に加算されます。 • 1 --- 基準値を下回る場合だけ：基準値を下回るピクセルのみが体積に加算されます。 • 2 --- 基準値を下回る場合は減算：基準値を上回るピクセルは体積に加算され、基準値を下回るピクセルは体積から減算されます。 • 3 --- 基準値を下回る場合は加算：ピクセル値が基準値より上か下かに関わらず、体積に加算します。これは、ピクセル値と基準値の差の絶対値による合計と等しいです。

次のページに続く

表 27.73 – 前のページからの続き

ラベル	名前	データ型	説明
体積計算のレポート	OUTPUT_HTML_FILE	[html] デフォルト：[一時ファイルに保存]	出力 HTML レポートを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
体積計算の表	OUTPUT_TABLE	[テーブル] デフォルト：[出力をスキップ]	出力テーブルを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
体積	VOLUME	[数値]	計算された体積
領域（Area）	AREA	[数値]	面積をマップ単位の 2 乗で表した値
ピクセル数	PIXEL_COUNT	[数値]	解析したピクセルの合計数
体積計算のレポート	OUTPUT_HTML_FILE	[html]	HTML 形式の出力レポート（体積、面積、ピクセル数を含む）
体積計算の表	OUTPUT_TABLE	[テーブル]	出力テーブル（体積、面積、ピクセル数を含む）

Python コード

Algorithm ID: native:rastersurfacevolume

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性テーブルによる再分類

ベクタテーブルで指定された値の範囲に基づいて新しいクラス値を割り当てて、ラスタバンドを再分類します。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	再分類するラスタレイヤ
バンド番号	RASTER_BAND	[ラスタのバンド] デフォルト：ラスタレイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、再分類したいバンドを選択してください。
クラス区分のテーブルを含むレイヤ	INPUT_TABLE	[ベクタ：任意]	分類に使用する値を含むベクタレイヤ
区分の下限を示す属性（フィールド）	MIN_FIELD	[テーブルのフィールド：数値]	クラス範囲の最小値を持つフィールド。最も低い値を取り込むには -inf を使います。
区分の上限を示す属性（フィールド）	MAX_FIELD	[テーブルのフィールド：数値]	クラス範囲の最大値を持つフィールド。最も高い値を取り込むには inf を使います。
クラスを示す属性（フィールド）	VALUE_FIELD	[テーブルのフィールド：数値]	そのクラス（対応する最小値と最大値の間）に当てはまるピクセルに割り当てる値を持つフィールド。その範囲の値を NoData に設定するには nan を使います。
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	NO_DATA	[数値] デフォルト：-9999.0	nodata 値に適用する値
分類区分の境界上の扱い	RANGE_BOUNDARIES	[列挙型] デフォルト：0	クラス分類の比較ルールを定義します。オプションは次のとおり： • 0 --- min < value <= max • 1 --- min <= value < max • 2 --- min <= value <= max • 3 --- min < value < max
値と一致する区分がない場合は nodata	NODATA_FOR_MISSI	[ブール値] デフォルト：False	どのクラスにも該当しないバンド値を nodata 値とします。False の場合には、元の値を保持します。
出力のデータ型	DATA_TYPE	[列挙型] デフォルト：5	出力ラスタファイルの形式を定義します。オプションは以下のとおり: • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 --- Int32 • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	再分類されたバンド値を持つ出力ラスタレイヤ

Python コード

Algorithm ID: native:reclassifybylayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

区分表（テーブル）で再分類

固定テーブルで指定された値の範囲に基づいて新しいクラス値を割り当てて、ラスタバンドを再分類します。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	再分類するラスタレイヤ
バンド番号	RASTER_BAND	[ラスタのバンド] デフォルト： 1	値を再計算したいラスタのバンド
再分類の区分表（テーブル）	TABLE	[テーブル]	各クラスの境界を設定する値（Minimum と Maximum）と、そのクラスに当てはまるバンド値に割り当てる新しい Value の 3 カラムのテーブル。値 -inf は最小値、inf は最大値として使用でき、nan は出力値を NoData に設定するために使用できます。

次のページに続く

表 27.76 – 前のページからの続き

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
nodata を出力する	NO_DATA	[数値] デフォルト：-9999.0	nodata 値に適用する値
分類区分の境界上の扱い	RANGE_BOUNDARIES	[列挙型] デフォルト：0	クラス分類の比較ルールを定義します。オプションは次のとおり： • 0 --- min < value <= max • 1 --- min <= value < max • 2 --- min <= value <= max • 3 --- min < value < max
値と一致する区分がない場合は nodata	NODATA_FOR_MISSI	[ブール値] デフォルト：False	どのクラスにも該当しないバンド値を nodata 値とします。False の場合には、元の値を保持します。

次のページに続く

表 27.77 – 前のページからの続き

ラベル	名前	データ型	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルの形式を定義します。オプションは以下のとおり: • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 --- Int32 • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	再分類されたバンド値を持つ出力ラスタレイヤ

Python コード

Algorithm ID: `native:reclassifybytable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタの範囲変更

ラスタのヒストグラム（ピクセル値）の形状（分布）は保ったまま、ラスタレイヤを再スケーリングして新しい値の範囲に変更します。入力値は、ソースラスタのピクセル最小値・最大値から出力結果のピクセル最小・最大範囲へと線形補間でマッピングされます。

デフォルトでは、このアルゴリズムは元の nodata 値をそのまま維持しますが、この動作を上書きするオプションがあります。

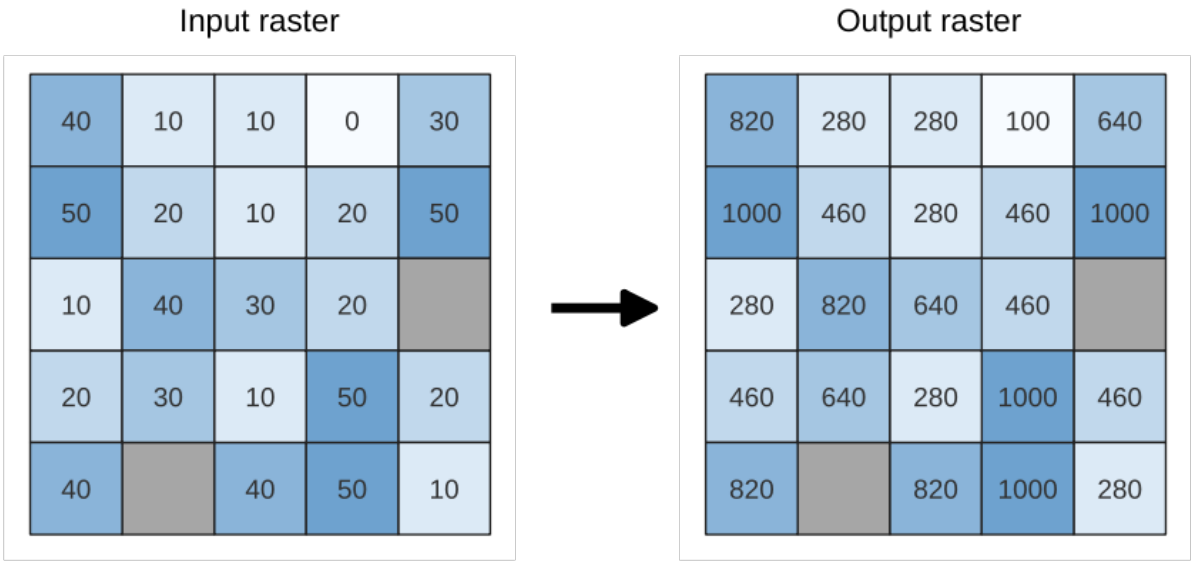


図 27.18: ラスタレイヤの値の範囲を [0 - 50] から [100 - 1000] へと再スケーリング

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT	[ラスタ]	値の範囲を変更するラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、バンドを選択してください。
最小値	MINIMUM	[数値] デフォルト値：0.0	範囲変更後のレイヤのピクセル最小値
最大値	MAXIMUM	[数値] デフォルト値：255.0	範囲変更後のレイヤのピクセル最大値
nodata 値 オプション	NODATA	[数値] デフォルト値：未設定	NODATA のピクセルに割り当てる値。未設定の場合には、元の NODATA 値が保持されます。
リスケール出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
リスケール出力	OUTPUT	[ラスタ]	範囲変更後のバンド値を持つ出力ラスタレイヤ

Python コード

アルゴリズム ID: native:rescaleraster

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタ値を四捨五入

ラスタデータセットのセル値を指定した小数点桁数で丸めます。

小数点以下とする代わりに、負の小数点以下桁数を使用して、値を基数 n の累乗に丸めることもできます。例えば、基数 n が 10 で小数点以下の桁数が-1 の場合、このアルゴリズムはセルの値を 10 の倍数に丸め、-2 の場合は 100 の倍数に丸めるといった具合です。任意の基数を選択しても、このアルゴリズムは同じ乗法の原理を適用します。セル値を基数 n の倍数に丸めることで、ラスタレイヤの値を簡略化することができます。

このアルゴリズムは、入力ラスタのデータ型を維持します。このため、byte 型や整数型のラスタは基数 n の倍数に丸めることしかできません。基数 n の倍数に丸められない場合には警告が発生し、byte 型や整数型のラスタとして、入力ラスタのコピーが返ります。

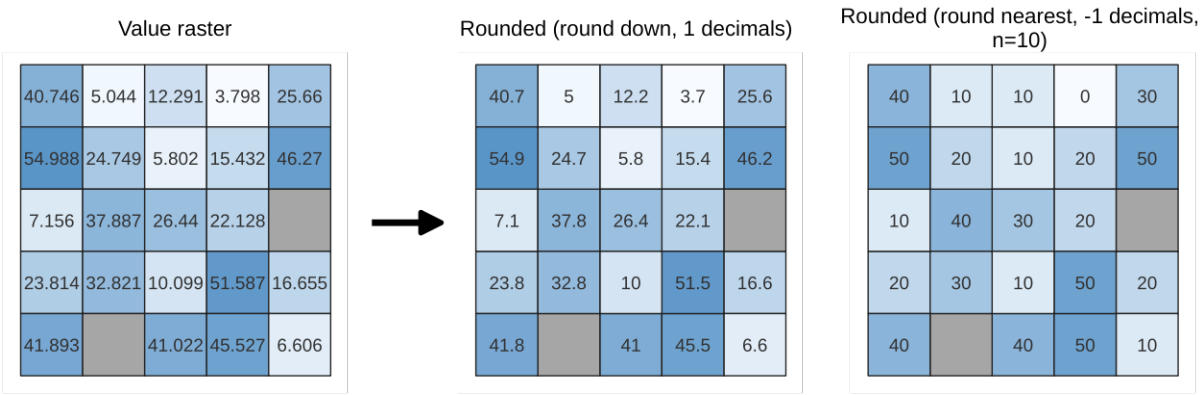


図 27.19: ラスタの値の丸め

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT	[ラスタ]	処理するラスタ
バンド番号	BAND	[数値] デフォルト： 1	ラスタのバンド番号
丸める方向	ROUNDING_DIRECTI	[リスト] デフォルト： 1	値を丸める方法。オプションは次のとおり： • 0 -- 繰り上げ • 1 -- 四捨五入 • 2 -- 切り下げ
小数点以下の桁数	DECIMAL_PLACES	[数値] デフォルト： 2	丸める小数点以下の桁数。負の値を使用すると、セルの値を基数 n の桁数で丸める

次のページに続く

表 27.78 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
基数	BASE_N	[数値] デフォルト： 10	DECIMAL_PLACES パラメータが負の場合は、ラスタの値は基数 n の桁数で丸められる

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	選択したバンドの値が丸められた出力ラスタレイヤ

Python コード

アルゴリズム ID: native:roundrastervalues

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤにラスタ値を付加

ポイントの位置におけるラスタ値を抽出します。ラスタレイヤがマルチバンドの場合には、各バンドの値がサンプリングされます。

結果レイヤの属性テーブルには、ラスタレイヤのバンド数分の新しい列が追加されます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	サンプリングに使用するポイントベクタレイヤ
ラスタレイヤ	RASTERCOPY	[ラスタ]	与えられたポイント位置でサンプリングされるラスタレイヤ
ラスタ値を収納するカラム名の接頭辞	COLUMN_PREFIX	[文字列] デフォルト：'SAMPLE_'	追加されるカラム名の接頭辞
サンプリングした出力 オプション	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	サンプリングされた値の出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
サンプリングした出力	OUTPUT	[ベクタ：ポイント]	サンプリングされた値の出力レイヤ

Python コード

Algorithm ID: native:rastersampling

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ゾーンヒストグラム

ポリゴン地物内に含まれるラスタレイヤの各ユニーク値の個数を表すフィールドを追加します。

出力レイヤの属性テーブルには、ポリゴンと交差するラスタレイヤのユニーク値の種類数分のフィールドが追加されます。

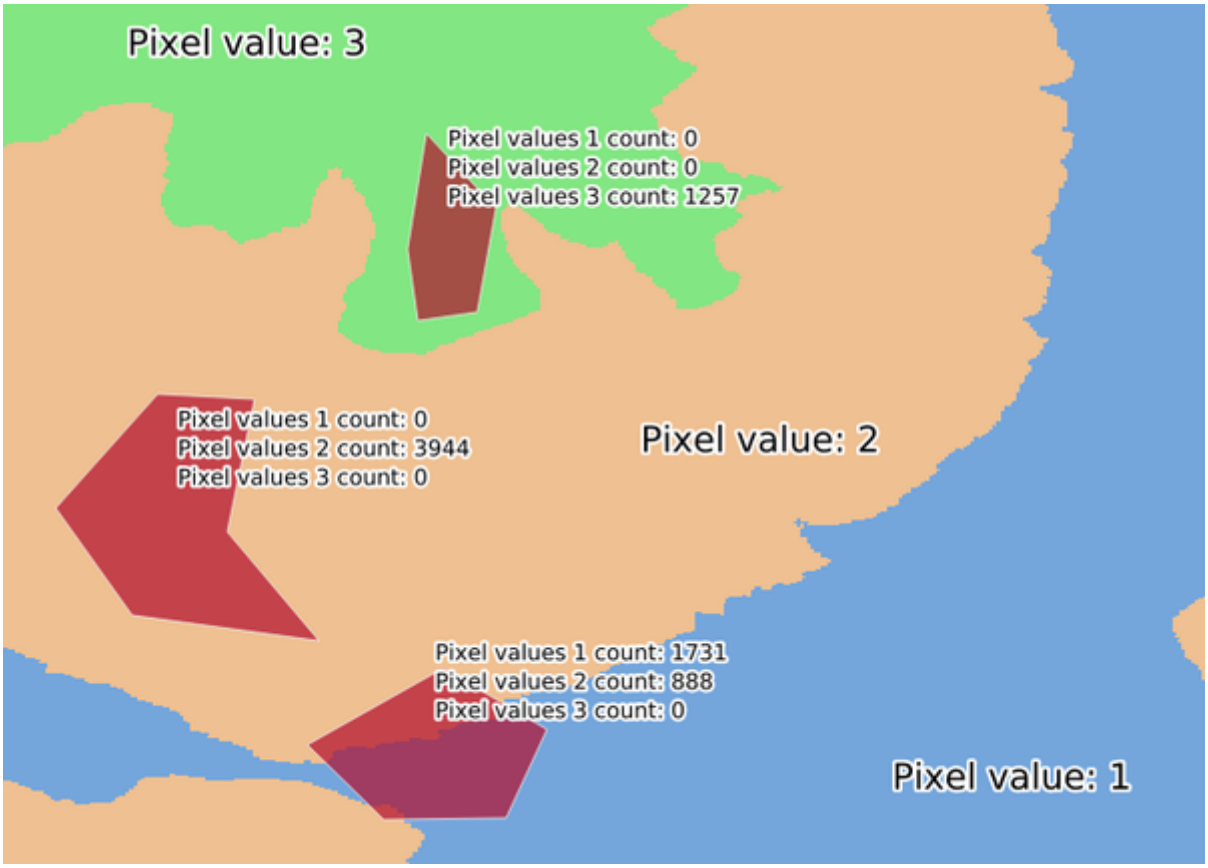


図 27.20: ラスタレイヤのヒストグラムの例

パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
バンド番号	RASTER_BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、バンドを選択してください。
ゾーンのベクタレイヤ	INPUT_VECTOR	[ベクタ：ポリゴン]	ゾーンを定義するベクタポリゴンレイヤ
ラスタ値を収納するカラム名の接頭辞	COLUMN_PREFIX オプション	[文字列] デフォルト：'HISTO_'	出力カラム名の接頭辞
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタポリゴンレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	出力ベクタポリゴンレイヤ

Python コード

Algorithm ID: native:zonalhistogram

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ゾーン統計量（ベクタ）

ラスタレイヤの統計量を、重なっているポリゴンベクタレイヤの地物ごとに計算します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	ゾーンを含むベクタポリゴンレイヤ
ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
対象バンド	RASTER_BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、統計量を計算したいバンドを選択してください。
ラスタ値を収納するカラム名の接頭辞	COLUMN_PREFIX	[文字列] デフォルト：'_'	出力カラム名の接頭辞
計算する統計量	STATISTICS	[列挙型] [リスト] デフォルト：[0,1,2]	出力する統計演算のリスト。オプションは以下のとおり： • 0 --- カウント（Count） • 1 --- 合計 • 2 --- 平均 • 3 --- 中央値 • 4 --- 標準偏差 • 5 --- 最小値 • 6 --- 最大 • 7 --- 範囲（Range） • 8 --- 最稀値（Minority） • 9 --- 最頻値（Majority） • 10 --- 種類（Variety） • 11 --- 分散（Variance）
ゾーン統計量出力	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタポリゴンレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ゾーン統計量出力	OUTPUT	[ベクタ：ポリゴン]	統計量が追加されたゾーンのベクタレイヤ

Python コード

Algorithm ID: `native:zonalstatisticsfb`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.12 ラスタ作成

定数ラスタ

指定された範囲とセルサイズに対して、指定値で埋められたラスタレイヤを作成します。

さらに、出力データ型を指定できます。選択した出力ラスタデータ型で表すことができない値が入力された場合、アルゴリズムは中止されます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	EXTENT	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	TARGET_CRS	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 0.1	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
定数値	NUMBER	[数値] デフォルト： 1	出力ラスタレイヤの定数ピクセル値。
定数ラスタ出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 5	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Byte • 1 --- Integer16 • 2 --- Unsigned Integer16 • 3 --- Integer32 • 4 --- Unsigned Integer32 • 5 --- Float32 • 6 --- Float64

出力

ラベル	名前	データ型	説明
定数ラスタ出力	OUTPUT	[ラスタ]	指定されたピクセルサイズと値を持つ、目的の範囲をカバーするラスタ。

Python コード

Algorithm ID: `native:createconstantrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（二項分布）

指定された範囲とセルサイズに対して、二項分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、N を 10、Probability を 0.5 とした場合の値が選択されます。これは、詳細パラメータの N と Probability を使用して上書きできます。ラスタのデータ型は整数型（デフォルトは Integer16）に設定されています。二項分布の乱数値は、正の整数として定義されています。浮動小数点ラスタは、整数値を浮動小数点へキャストしたものです。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	EXTENT	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	TARGET_CRS	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 0.1	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Integer16 • 1 --- Unsigned Integer16 • 2 --- Integer32 • 3 --- Unsigned Integer32 • 4 --- Float32 • 5 --- Float64
N	N	[数値] デフォルト： 10	
Probability	PROBABILITY	[数値] デフォルト： 0.5	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: `native:createrandombinomialrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（指数分布）

指定された範囲とセルサイズに対して、指数分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、Lambda を 1.0 とした場合の値が選択されます。これは、詳細パラメータの Lambda を使用して上書きできます。指数分布の乱数値は浮動小数点であるため、ラスタのデータ型はデフォルトで Float32 に設定されています。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	`EXTENT`	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	`TARGET_CRS`	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	`PIXEL_SIZE`	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	`OUTPUT`	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Float32 • 1 --- Float64
Lambda	LAMBDA	[数値] デフォルト： 1.0	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: `native:createrandomexponentialrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（ガンマ分布）

指定された範囲とセルサイズに対して、ガンマ分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、Alpha、Beta とも 1.0 とした場合の値が選択されます。これは、詳細パラメータの Alpha と Beta を使用して上書きできます。ガンマ分布の乱数値は浮動小数点であるため、ラスタのデータ型はデフォルトで Float32 に設定されています。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	EXTENT	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	TARGET_CRS	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Float32 • 1 --- Float64
Alpha	ALPHA	[数値] デフォルト： 1.0	
Beta	BETA	[数値] デフォルト： 1.0	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: `native:createrandomgammarasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（幾何分布）

指定された範囲とセルサイズに対して、幾何分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、Probability を 0.5 とした場合の値が選択されます。これは、詳細パラメータの Probability を使用して上書きできます。ラスタのデータ型は整数型（デフォルトは Integer16）に設定されています。幾何分布の乱数値は、正の整数として定義されています。浮動小数点ラスタは、整数値を浮動小数点へキャストしたものです。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	EXTENT	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	TARGET_CRS	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Integer16 • 1 --- Unsigned Integer16 • 2 --- Integer32 • 3 --- Unsigned Integer32 • 4 --- Float32 • 5 --- Float64
Probability	PROBABILITY	[数値] デフォルト： 0.5	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: `native:createrandomgeometricrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（負の二項分布）

指定された範囲とセルサイズに対して、負の二項分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、Distribution parameter k を 10.0、Probability を 0.5 とした場合の値が選択されます。これは、詳細パラメータの k と Probability を使用して上書きできます。ラスタのデータ型は整数型（デフォルトは Integer16）に設定されています。負の二項分布の乱数値は、正の整数として定義されています。浮動小数点ラスタは、整数値を浮動小数点へキャストしたものです。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	`EXTENT`	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	`TARGET_CRS`	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	`PIXEL_SIZE`	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	`OUTPUT`	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Integer16 • 1 --- Unsigned Integer16 • 2 --- Integer32 • 3 --- Unsigned Integer32 • 4 --- Float32 • 5 --- Float64
Distribution parameter k	K_PARAMETER	[数値] デフォルト： 10	
Probability	PROBABILITY	[数値] デフォルト： 0.5	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: `native:createrandomnegativebinomialrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（正規分布）

指定された範囲とセルサイズに対して、正規分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、平均 0.0、標準偏差 1.0 とした場合の値が選択されます。これは、詳細パラメータの平均値と標準偏差を使用して上書きできます。正規分布の乱数値は浮動小数点であるため、ラスタのデータ型はデフォルトで Float32 に設定されています。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	`EXTENT`	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	`TARGET_CRS`	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	`PIXEL_SIZE`	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	`OUTPUT`	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Float32 • 1 --- Float64
Mean of normal distribution	MEAN	[数値] デフォルト： 0.0	
Standard deviation of normal distribution	STDDEV	[数値] デフォルト： 1.0	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: native:createrandomnormalrasterlayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（ポワソン分布）

指定された範囲とセルサイズに対して、ポアソン分布の乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、Mean を 1.0 とした場合の値が選択されます。これは、詳細パラメータの Mean を使用して上書きできます。ラスタのデータ型は整数型（デフォルトは Integer16）に設定されています。ポアソン分布の乱数値は、正の整数として定義されています。浮動小数点ラスタは、整数値を浮動小数点へキャストしたものです。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	EXTENT	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	TARGET_CRS	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	PIXEL_SIZE	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 0	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Integer16 • 1 --- Unsigned Integer16 • 2 --- Integer32 • 3 --- Unsigned Integer32 • 4 --- Float32 • 5 --- Float64
Mean	MEAN	[数値] デフォルト： 1.0	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: native:createrandompoissonrasterlayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

乱数ラスタの作成（一様分布）

指定された範囲とセルサイズに対して、乱数値で埋められたラスタレイヤを作成します。

デフォルトでは、値は指定された出力ラスタタイプの最小値と最大値の範囲になります。これは、詳細パラメータの下限値と上限値を使用して上書きできます。境界の値が同じであるか、両方がゼロ（デフォルト）の場合、アルゴリズムは、選択したラスタデータ型の値の全範囲でランダムな値を作成します。出力ラスタタイプの許容範囲外の境界を選択すると、アルゴリズムは中止されます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	`EXTENT`	[範囲]	出力されるラスタレイヤの空間的範囲を指定します。内部的にはタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
ラスタの CRS	`TARGET_CRS`	[crs] デフォルト：プロジェクト CRS	出力ラスタレイヤの CRS
ピクセルサイズ	`PIXEL_SIZE`	[数値] デフォルト： 1.0	地図単位であらわしたピクセルサイズ (X=Y)。最小値：0.01
出力ラスタ	`OUTPUT`	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
出力データ型	OUTPUT_TYPE デフォルト： 5	[列挙型]	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 --- Byte • 1 --- Integer16 • 2 --- Unsigned Integer16 • 3 --- Integer32 • 4 --- Unsigned Integer32 • 5 --- Float32 • 6 --- Float64
Lower bound for random number range	LOWER_BOUND	[数値] デフォルト： 0.0	
Upper bound for random number range	UPPER_BOUND	[数値] デフォルト： 0.0	

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	指定されたセルサイズで目的の範囲をカバーする、乱数値で埋められたラスタ

Python コード

Algorithm ID: `native:createrandomuniformrasterlayer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.13 ラスタ地形解析

傾斜方位（aspect）

入力された数値地形モデルの傾斜方位を計算します。結果の方位ラスタレイヤには 0 から 360 までの値が含まれており、これは北（0 °）から時計回りの斜面の方向を表します。

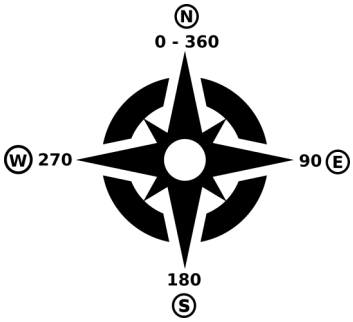


図 27.21: 傾斜方位の値

以下の画像は、傾斜方位のレイヤをカラーランプを使用して分類したものです。

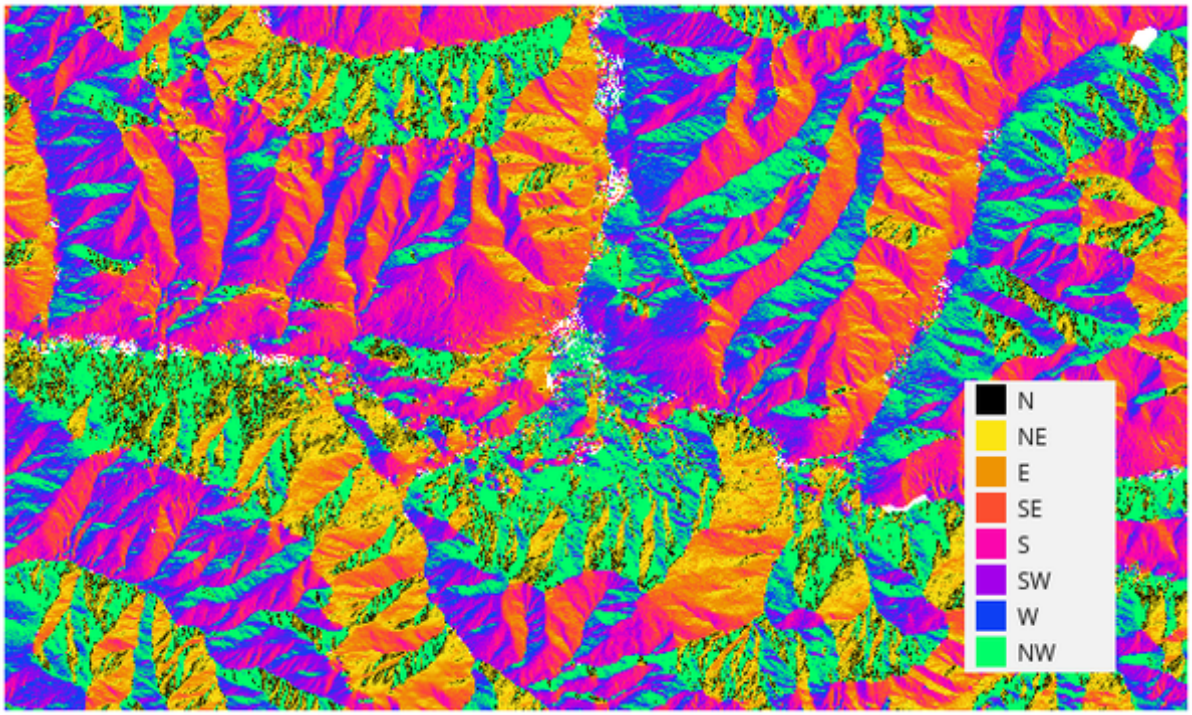


図 27.22: 分類された傾斜方位レイヤ

パラメータ

ラベル	名前	データ型	説明
DEM レイヤ	INPUT	[ラスタ]	数値地形モデルのラスタレイヤ
Z 係数	Z_FACTOR	[数値] デフォルト： 1.0	鉛直方向の拡大率。このパラメータは、Z の単位が XY の単位と異なる場合、例えばフィートとメートルの場合に有効です。このパラメータを使って Z を調整することができます。デフォルトは 1（拡大なし）です。
傾斜方位（aspect）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	傾斜方位の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
傾斜方位（aspect）	OUTPUT	[ラスタ]	傾斜方位の出力ラスタレイヤ

Python コード

Algorithm ID: `qgis:aspect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

陰影図（hillshade）

入力された数値地形モデルの陰影図ラスタレイヤを計算します。

レイヤの陰影付けは、太陽の位置に応じて計算されます。太陽の水平方向の角度（方位角）と垂直方向の角度（太陽高）の両方を変更するオプションがあります。

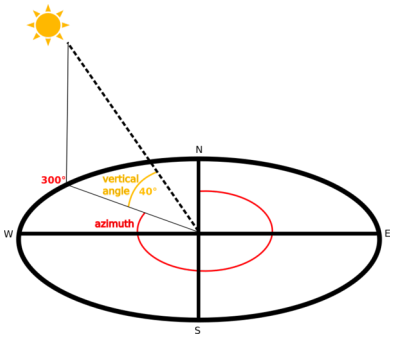


図 27.23: 方位角と太陽高

陰影図レイヤには、0（完全な日陰）から 255（完全な日向）までの値が含まれます。陰影図は通常、その領域の起伏をわかりやすくするために使用されます。

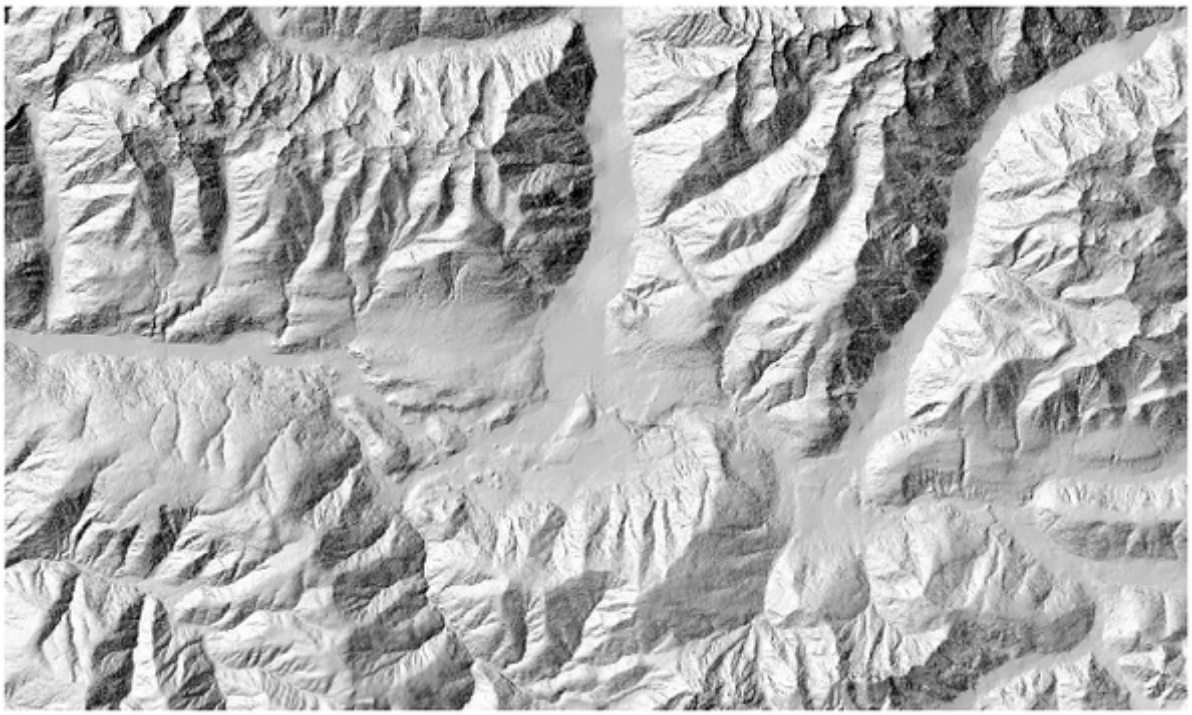


図 27.24: 方向角 300 度、太陽光 45 度とした陰影図

陰影図レイヤに透明度の値を与え標高ラスタと重ね合わせると、特に興味深い図ができあがります。

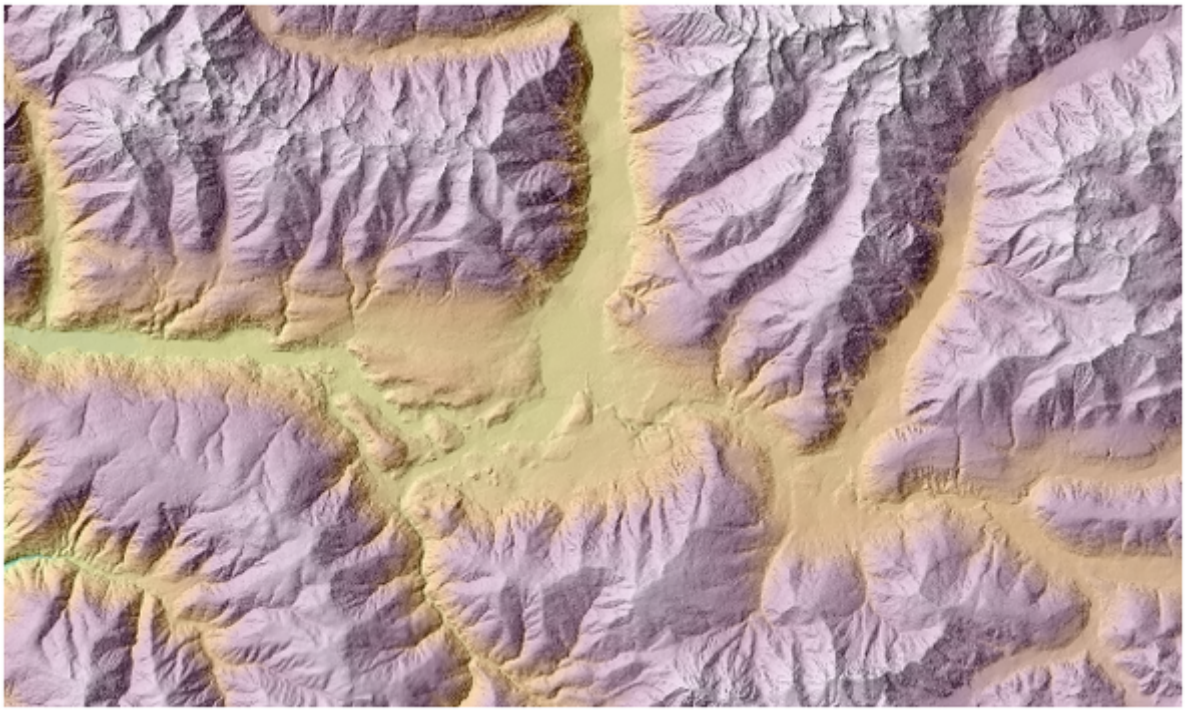


図 27.25: 陰影図と標高レイヤの重ね合わせ図

パラメータ

ラベル	名前	データ型	説明
DEM レイヤ	INPUT	[ラスタ]	数値地形モデルのラスタレイヤ
Z 係数	Z_FACTOR	[数値] デフォルト： 1.0	鉛直方向の拡大率。このパラメータは、Z の単位が XY の単位と異なる場合、例えばフィートとメートルの場合に有効です。このパラメータを使って Z を調整することができます。このパラメータ値を増加させると、最終結果が強調されます（より「でこぼこ」して見えるようになります）。デフォルトは 1（拡大なし）です。
方位角（Azimuth）	AZIMUTH	[数値] デフォルト：300.0	太陽の水平方向角度（度単位、時計回り方向）を設定します。値の範囲は 0 から 360 までです。北は 0 度です。
太陽高	V_ANGLE	[数値] デフォルト： 40.0	太陽の鉛直角度（度単位）、つまり太陽の高度を設定します。0（最小高さ）から 90（最大高さ）までの値を取ります。

次のページに続く

表 27.81 – 前のページからの続き

ラベル	名前	データ型	説明
陰影図（hillshade）	OUTPUT	[ラスタ] デフォルト：一時ファイルに保存	陰影図の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
陰影図（hillshade）	OUTPUT	[ラスタ]	陰影図の出力ラスタレイヤ

Python コード

Algorithm ID: `qgis:hillshade`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

高度面積曲線（hypsometric）

入力された DEM の高度面積曲線を計算します。計算結果は指定された出力フォルダに CSV テーブルとして出力されます。

高度面積曲線とは、ある地理的地域の標高値の累積ヒストグラムのことです。

高度面積曲線を使って、土地の地形による景観の違いを見つけることができます。

パラメータ

ラベル	名前	データ型	説明
DEM レイヤ	INPUT_DEM	[ラスタ]	標高の計算に使用する数値地形モデルラスタレイヤ

次のページに続く

表 27.82 – 前のページからの続き

ラベル	名前	データ型	説明
境界レイヤ	BOUNDARY_LAYER	[ベクタ：ポリゴン]	高度面積曲線の計算に使用する領域の境界のポリゴンベクタレイヤ
ステップ	STEP	[数値] デフォルト：100.0	曲線の鉛直間隔
絶対値ではなく割合（%）を使用する	USE_PERCENTAGE	[ブール値] デフォルト：False	面積の絶対値ではなく、CSV ファイルの「Area」フィールドに面積のパーセンテージを書き出します
高度面積曲線（hypsometric）	OUTPUT_DIRECTORY	[フォルダ]	高度面積曲線の出力フォルダを指定します。次のいずれかです： • 一時ディレクトリに保存 • ディレクトリに保存します

出力

ラベル	名前	データ型	説明
高度面積曲線（hypsometric）	OUTPUT_DIRECTORY	[フォルダ]	面積高度曲線のファイルが保存されるディレクトリ。入力ベクタレイヤの各地物について、面積と高度の CSV ファイルが作成されます。 ファイル名は histogram_ で始まり、レイヤ名と地物 ID がこれに続きます。

	A	B
1	Area	Elevation
2	177475194.383	307
3	233206029.24	407
4	295553735.793	507
5	394718815.615	607
6	501801102.615	707
7	624399019.792	807
8	828877274.39	907
9	1042693465.68	1007
10	1277373021.81	1107
11	1556443975.41	1207
12	1888617494.27	1307
13	2248520437.31	1407
14	2627916813.17	1507
15	3010880212.04	1607
16	3411087555.34	1707

Python コード

Algorithm ID: `qgis:hypsometriccurves`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

段彩図（relief）

数値標高データから影付きの段彩図レイヤを作成します。段彩の色を手動で指定することもできますが、アルゴリズムにすべてのクラスを自動的に選択させることもできます。

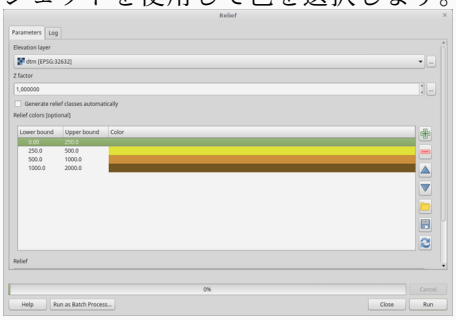
図 27.26: 段彩図レイヤ

パラメータ

ラベル	名前	データ型	説明
DEM レイヤ	INPUT	[ラスタ]	数値地形モデルのラスタレイヤ
Z 係数	Z_FACTOR	[数値] デフォルト： 1.0	鉛直方向の拡大率。このパラメータは、Z の単位が XY の単位と異なる場合、例えばフィートとメートルの場合に有効です。このパラメータを使って Z を調整することができます。このパラメータ値を増加させると、最終結果が強調されます（より「でこぼこ」して見えるようになります）。デフォルトは 1（拡大なし）です。
段彩の区分を自動的に生成	AUTO_COLORS	[ブール値] デフォルト： False	このオプションにチェックを入れると、アルゴリズムが全ての段彩色クラスを自動的に作成します

次のページに続く

表 27.83 – 前のページからの続き

ラベル	名前	データ型	説明
段彩の色 オプション	COLORS	[テーブルウィジェット]	段彩の色を手動で選択したい場合には、このテーブルウィジェットを使用します。色クラスを好きな数だけ追加し、各クラスについて上下の境界値を選択し、色の行をクリックして色ウィジェットを使用して色を選択します。  図 27.27: 段彩の色クラスの手動設定 パネルの右横にあるボタンを使用すると、次の操作ができます：色クラスの追加と削除、定義済みの色クラスの並べ替え、色クラスが定義された既存のファイルを開く、現在の色クラスをファイルとして保存する
段彩図（relief）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	段彩図の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...
頻度分布の出力 オプション	FREQUENCY_DISTRI	[テーブル] デフォルト：[出力をスキップ]	頻度分布の出力 CSV テーブルを指定します。次のいずれかです： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
段彩図（relief）	OUTPUT	[ラスタ]	段彩図の出力ラスタレイヤ
頻度分布の出力	OUTPUT	[テーブル]	頻度分布の出力結果

Python コード

Algorithm ID: `qgis:relief`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地形起伏指数（TRI）

Riley et al. (1999) による、地形の不均一性に関する定量的な指標値を計算します。これは地点ごとに計算される、3x3 ピクセルのグリッド内の標高変化を集計した値です。

各ピクセルには、中央のセルとそれを囲む 8 セルの間の標高差の値が含まれます。

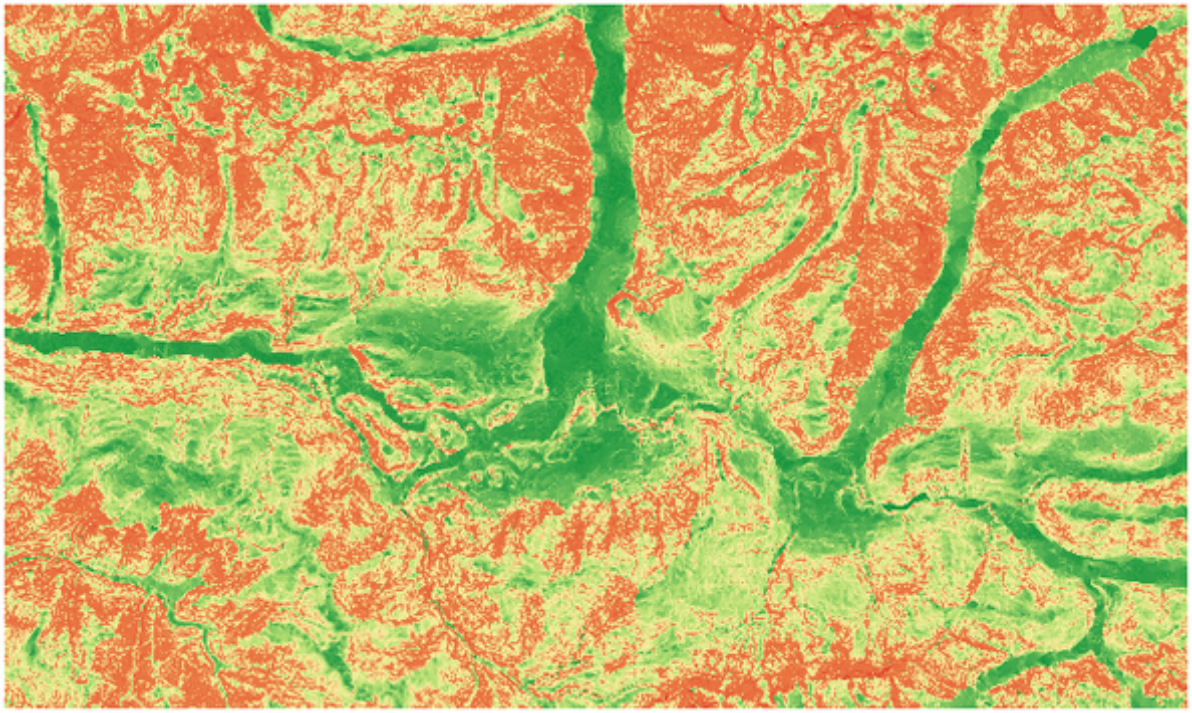


図 27.28: 赤（指数値小）から緑（指数値大）で表示した地形起伏指数レイヤ

パラメータ

ラベル	名前	データ型	説明
DEM レイヤ	INPUT	[ラスタ]	数値地形モデルのラスタレイヤ
Z 係数	Z_FACTOR	[数値] デフォルト： 1.0	鉛直方向の拡大率。このパラメータは、Z の単位が XY の単位と異なる場合、例えばフィートとメートルの場合に有効です。このパラメータを使って Z を調整することができます。このパラメータ値を増加させると、最終結果が強調されます（より「でこぼこ」して見えるようになります）。デフォルトは 1（拡大なし）です。
起伏指数出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	地形起伏指数の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
起伏指数出力	OUTPUT	[ラスタ]	地形起伏指数の出力ラスタレイヤ

Python コード

Algorithm ID: `qgis:ruggednessindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

傾斜（slope）

入力ラスタレイヤの傾斜を計算します。傾斜とは地形の傾きの角度のことで、 **度** 単位で表されます。

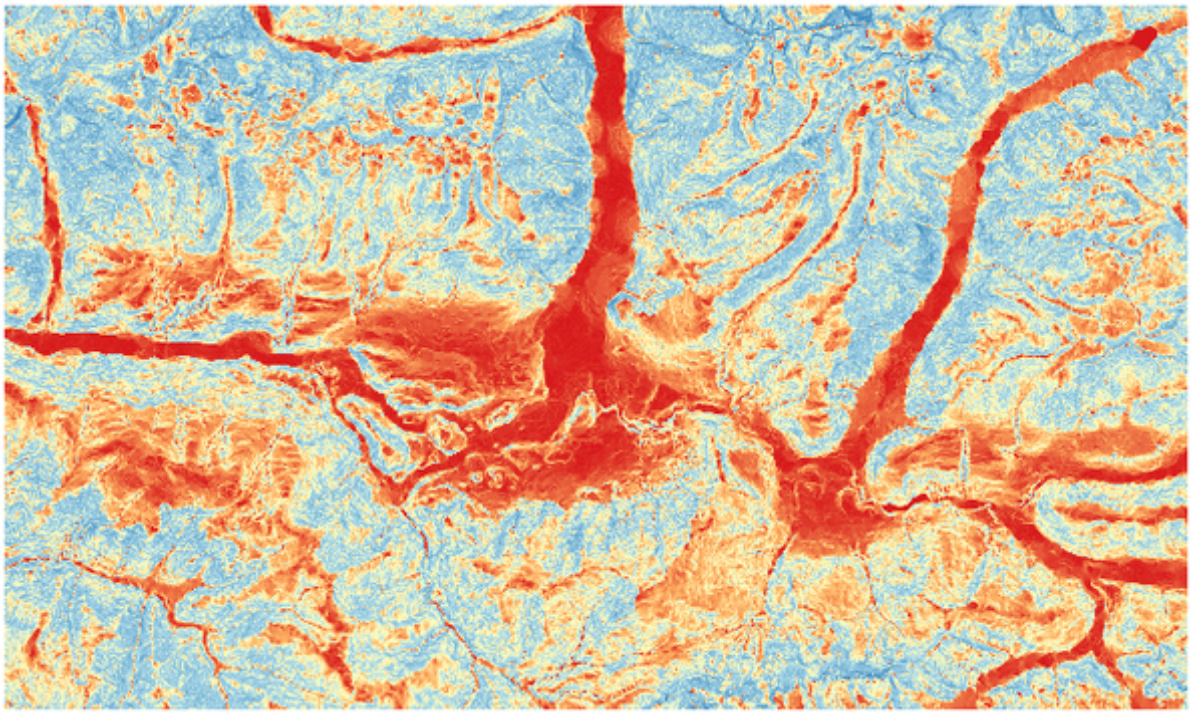


図 27.29: 平坦な場所は赤、急峻な場所は青で表示した傾斜レイヤ

パラメータ

ラベル	名前	データ型	説明
DEM レイヤ	INPUT	[ラスタ]	数値地形モデルのラスタレイヤ
Z 係数	Z_FACTOR	[数値] デフォルト： 1.0	鉛直方向の拡大率。このパラメータは、Z の単位が XY の単位と異なる場合、例えばフィートとメートルの場合に有効です。このパラメータを使って Z を調整することができます。このパラメータ値を増加させると、最終結果が強調されます（より急峻に見えるようになります）。デフォルトは 1（拡大なし）です。
傾斜（slope）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	傾斜の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
傾斜（slope）	OUTPUT	[ラスタ]	傾斜の出力ラスタレイヤ

Python コード

Algorithm ID: `qgis:slope`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.14 ラスタツール

地図のラスタ化

マップキャンバスの内容のラスタ画像を作成します。

地図テーマ を選択して、予め指定されたレイヤの組み合わせと各レイヤの定義済みスタイルでレンダリングすることができます。

地図テーマを使用しない場合には、単一のレイヤを選択することもできます。

地図テーマもレイヤも設定されていない場合には、現在の地図コンテンツがレンダリングされます。入力された最小範囲は内部的にタイルサイズの倍数に拡張されます。

パラメータ

ラベル	名前	データ型	説明
書き出す範囲 (xmin, xmax, ymin, ymax)	EXTENT	[範囲]	出力ラスタレイヤの範囲を指定します。内部的にタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
タイルサイズ	TILE_SIZE	[数値] デフォルト： 1024	出力ラスタレイヤのタイルサイズ。最小値：64
ピクセルの地図単位	MAP_UNITS_PER_PI	[数値] デフォルト：100.0	（地図単位の）ピクセルのサイズ。最小値：0.0
背景を透明にする	MAKE_BACKGROUND_	[ブール値] デフォルト： False	透明な背景で地図をエクスポートします。 `True` に設定された場合、（RGB ではなく）RGBA 画像で出力します。
地図のテーマ オプション	MAP_THEME	[列挙型]	レンダリング時に既存の **地図テーマ** を使用します。
書き出すレイヤ オプション	LAYER	[列挙型]	書き出すレイヤを選択します
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ラスタ]	出力ラスタレイヤ

Python コード

Algorithm ID: native:rasterize

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

nodata セルを値で埋める

入力ラスタの nodata 値をリセットし、指定した値に設定します。この結果、nodata ピクセルの無いラスタデータセットが作成されます。

このアルゴリズムは入力ラスタのデータ型に従います。例えば、整数値ラスタを埋める値に浮動小数点値を適用しても、値は丸められます。

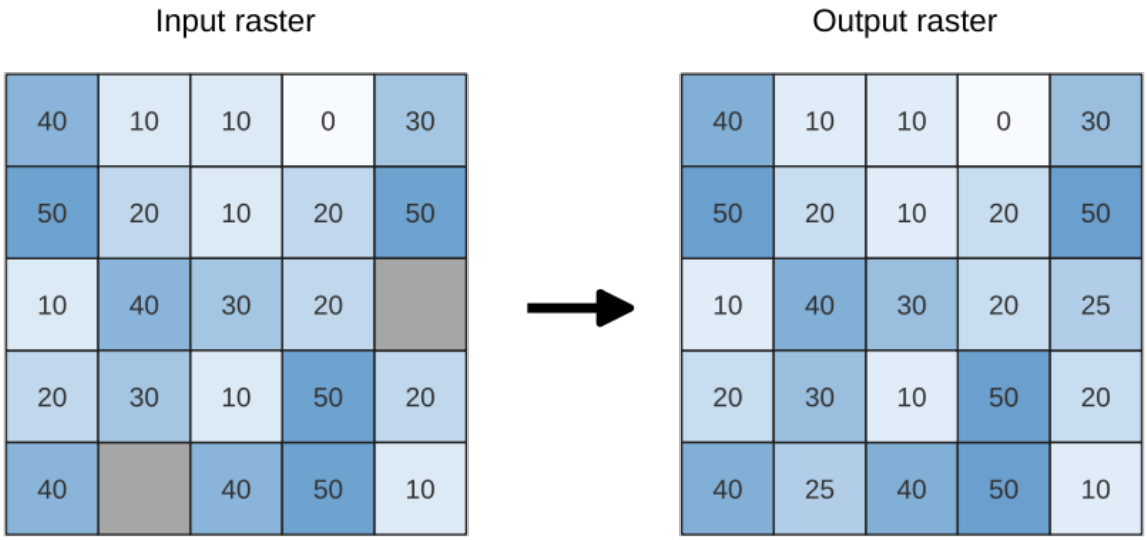


図 27.30: ラスタの nodata 値（グレー色）を埋める

パラメータ

ラベル	名前	データ型	説明
入力ラスタ	INPUT	[ラスタ]	処理するラスタ
バンド番号	BAND	[数値] デフォルト： 1	ラスタのバンド番号
埋める値	FILL_VALUE	[数値] デフォルト： 1.0	nodata ピクセルに使用する値を設定します
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ラスタ	OUTPUT	[ラスタ]	nodata セルが産められた出力ラスタレイヤ

Python コード

Algorithm ID: `native:fillnodata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

XYZ タイルの生成（ディレクトリ形式）

現在の QGIS プロジェクトを使用して“ XYZ ”ラスタタイルを生成します。個々のタイル画像はディレクトリ構造で保存されます。

パラメータ

ラベル	名前	データ型	説明
範囲 (xmin, xmax, ymin, ymax)	`EXTENT`	[範囲]	タイルの範囲を指定します。内部的にタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
最小ズーム値	`ZOOM_MIN`	[数値] デフォルト： 12	最小値 0、最大値 25
最大ズーム値	`ZOOM_MAX`	[数値] デフォルト： 12	最小値 0、最大値 25
DPI	`DPI`	[数値] デフォルト： 96	最小値 48、最大値 600
背景色 オプション	`BACKGROUND_COLOR`	[色] デフォルト： QColor(0, 0, 0, 0)	タイルの背景色を選択します
タイル形式	`TILE_FORMAT`	[列挙型] デフォルト： 0	次のいずれかです： • 0 --- PNG • 1 --- JPG
品質 (JPG の場合) オプション	`QUALITY`	[数値] デフォルト： 75	最小値 1、最大値 100

次のページに続く

表 27.85 – 前のページからの続き

ラベル	名前	データ型	説明
メタタイルのサイズ オプション	METATILESIZE	[数値] デフォルト： 4	XYZ タイルを生成する際のカスタムメタタイルサイズを指定します。大きな値を指定するとタイルのレンダリングが高速化され、ラベリングが向上する（ラベルのない隙間が少なくなる）可能性がありますが、メモリ使用量は増加します。最小値は 1、最大値は 20 です。
タイルの幅 オプション	TILE_WIDTH	[数値] デフォルト： 256	最小値 1、最大値 4096
タイルの高さ オプション	TILE_HEIGHT	[数値] デフォルト： 256	最小値 1、最大値 4096
Y 軸の反転（TMS） オプション	TMS_CONVENTION	[ブール値] デフォルト： False	
出力フォルダ オプション	OUTPUT_DIRECTORY	[フォルダ] デフォルト：[一時フォルダに保存]	（タイルの）出力ディレクトリを指定します。次のいずれかです： • 出力をスキップ • 一時ディレクトリに保存 • ディレクトリに保存します
html ファイル（Leaflet） オプション	OUTPUT_HTML	[html] デフォルト：[一時ファイルに保存]	HTML 出力ファイルを指定します。次のいずれかです： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力フォルダ	OUTPUT_DIRECTORY	[フォルダ]	（タイルの）出力ディレクトリ
html ファイル（Leaflet）	OUTPUT_HTML	[html]	HTML（Leaflet）の出力ファイル

Python コード

Algorithm ID: `qgis:tilesxyzdirectory`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

XYZ タイルの生成（MBTiles 形式）

現在の QGIS プロジェクトを使用して " XYZ " ラスタタイルを生成します。タイル画像は単一の " MBTiles " 形式ファイルとして保存されます。

パラメータ

ラベル	名前	データ型	説明
範囲 (xmin, xmax, ymin, ymax)	`EXTENT`	[範囲]	タイルの範囲を指定します。内部的にタイルサイズの倍数に拡張されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
最小ズーム値	`ZOOM_MIN`	[数値] デフォルト： 12	最小値 0、最大値 25

次のページに続く

表 27.86 – 前のページからの続き

ラベル	名前	データ型	説明
最大ズーム値	ZOOM_MAX	[数値] デフォルト： 12	最小値 0、最大値 25
DPI	DPI	[数値] デフォルト： 96	最小値 48、最大値 600
背景色 オプション	BACKGROUND_COLOR	[色] デフォルト： QColor(0, 0, 0, 0)	タイルの背景色を選択します
タイル形式	TILE_FORMAT	[列挙型] デフォルト： 0	次のいずれかです： • 0 --- PNG • 1 --- JPG
品質（JPG の場合） オプション	QUALITY	[数値] デフォルト： 75	最小値 1、最大値 100
メタタイルのサイズ オプション	METATILESIZE	[数値] デフォルト： 4	XYZ タイルを生成する際のカスタムメタタイルサイズを指定します。大きな値を指定するとタイルのレンダリングが高速化され、ラベリングが向上する（ラベルのない隙間が少なくなる）可能性がありますが、メモリ使用量は増加します。最小値は 1、最大値は 20 です。
出力ファイル（MBTiles 形式）	OUTPUT_FILE	[ファイル] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力ファイル（MBTiles 形式）	OUTPUT_FILE	[ファイル]	出力ファイル

Python コード

Algorithm ID: `qgis:tilesxyzmbtiles`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.15 ベクタ解析

属性の基本統計量

ベクタレイヤの属性テーブルのフィールドについて、基本統計量を計算します。

数値型、日付型、時間型、文字列型のフィールドを対象とします。

フィールドの型によって、計算される統計量は異なります。

統計量の結果は HTML ファイルとして作成され、 **プロセシング　結果ビューア** からアクセスできます。

デフォルトメニュー: **ベクタ　解析ツール**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT_LAYER	[ベクタ：任意]	統計量を計算するベクタレイヤ
統計量を計算する属性（フィールド）	FIELD_NAME	[テーブルのフィールド：任意]	統計量の計算に対応した任意のテーブルのフィールド
統計量の出力 オプション	OUTPUT_HTML_FILE	[html] デフォルト：[一時ファイルに保存]	計算した統計を出力するファイルの指定。次のいずれかです: • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
統計量の出力	OUTPUT_HTML_FILE	[html]	統計量の計算結果の HTML ファイル
カウント（Count）	COUNT	[数値]	
ユニークな値の種類	UNIQUE	[数値]	
空白値（null）の数	EMPTY	[数値]	
非空白値の数	FILLED	[数値]	
最小値（Minimum）	MIN	[入力レイヤと同じ]	
最大値（Maximum）	MAX	[入力レイヤと同じ]	
最短長（Min Length）	MIN_LENGTH	[数値]	
最大長（Max Length）	MAX_LENGTH	[数値]	
平均長（Mean Length）	MEAN_LENGTH	[数値]	
分散係数（Coef of Variance）	CV	[数値]	
合計（Sum）	SUM	[数値]	
平均値（Mean）	MEAN	[数値]	
標準偏差（Standard deviation）	STD_DEV	[数値]	
範囲（Range）	RANGE	[数値]	
中央値（Median）	MEDIAN	[数値]	
最稀値（rarest occurring value）	MINORITY	[入力レイヤと同じ]	
最頻値（most frequently occurring value）	MAJORITY	[入力レイヤと同じ]	
第１四分位（Q1）	FIRSTQUARTILE	[数値]	
第３四分位（Q3）	THIRDQUARTILE	[数値]	
四分位範囲（IQR）	IQR	[数値]	

Python コード

Algorithm ID: `qgis:basicstatisticsforfields`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線に沿った上昇下降量

ラインジオメトリに沿った総上昇量と総下降量を計算します。入力レイヤは Z 値を持っていなければなりません。レイヤが Z 値を持っていない場合には、 **ドレープ（ラスタ値を Z 値に代入）** アルゴリズムを使用して DEM レイヤから Z 値を付加できます。

出力レイヤは入力レイヤのコピーですが、各ラインジオメトリについて総上昇量 (`climb`) 、総下降量 (`descent`) 、最低標高 (`minelev`) 、最高標高 (`maxelev`) のフィールドが追加されたものです。もし入力レイヤがこれらの追加フィールドと同名のフィールドを持つ場合には、追加されるフィールドは（重複したものについて順に、 "name_2"、 "name_3" 等に）変更されます。

パラメータ

ラベル	名前	データ型	説明
線レイヤ	`INPUT`	[ベクタ：ライン]	上昇下降量を計算するラインレイヤ。Z 値を持っている必要があります
上昇下降量の出力レイヤ	`OUTPUT`	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力（ライン）レイヤの指定。次のいずれかです: • 一時レイヤを作成（ `TEMPORARY_OUTPUT` ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
上昇下降量の出力レイヤ	`OUTPUT`	[ベクタ：ライン]	上昇下降量計算の結果の新しい属性を持ったラインレイヤ
総上昇量	`TOTALCLIMB`	[数値]	入力レイヤの全てのラインジオメトリの上昇量の合計
総下降量	`TOTALDESCENT`	[数値]	入力レイヤの全てのラインジオメトリの下降量の合計
最低標高	`MINELEVATION`	[数値]	レイヤ内のジオメトリの最小標高値
最高標高	`MAXELEVATION`	[数値]	レイヤ内のジオメトリの最高標高値

Python コード

Algorithm ID: `qgis:climbalongline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ポリゴン内の点の数

ポイントレイヤとポリゴンレイヤを引数にとり、ポリゴンレイヤの各ポリゴン内にあるポイントレイヤのポイントの数を数えます。

入力ポリゴンレイヤと同じ内容の新しいポリゴンレイヤが作成されますが、これに加えて、各ポリゴンに対応したポイントの個数のフィールドが追加されています。

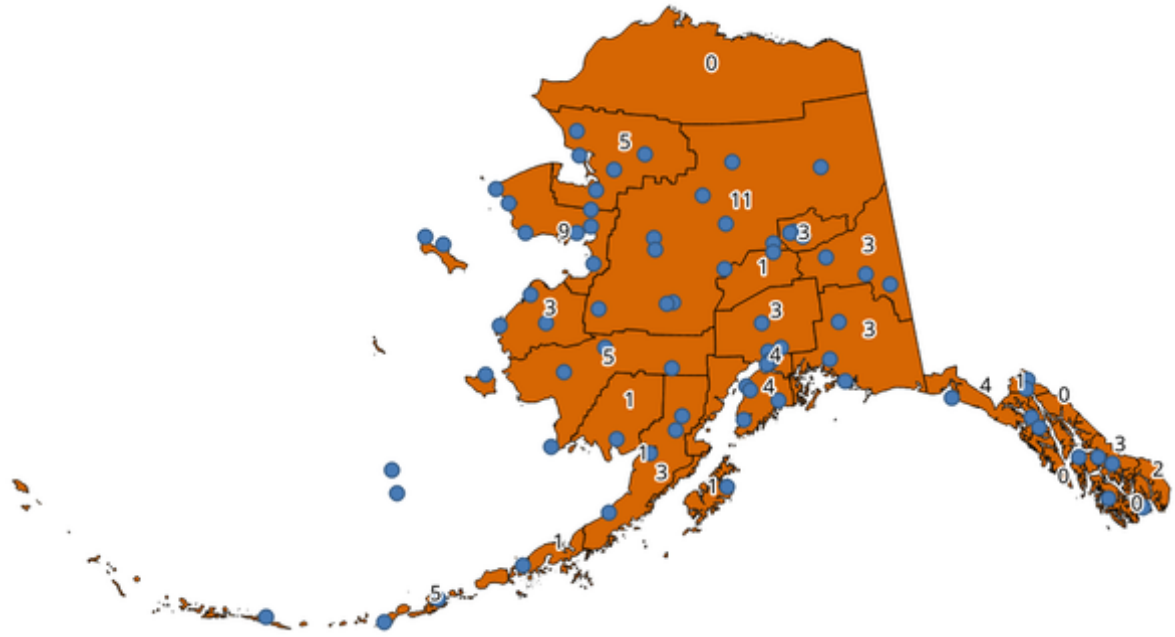


図 27.31: ポイントの数を表示したポリゴンのラベル

オプションの重み属性を使用すると、各ポイントに重みを割り当てることができます。また、ユニークな分類属性を指定することもできます。両方のオプションが使用された場合、重み属性が優先され、ユニークな分類属性は無視されます。

ポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー: **ベクタ　解析ツール**

パラメータ

ラベル	名前	データ型	説明
ポリゴン（Polygons）	POLYGONS	[ベクタ：ポリゴン]	地物内のポイントを数えるポリゴンレイヤ
ポイント（Points）	POINTS	[ベクタ：ポイント]	地物数を数えたいポイントレイヤ
重み属性（フィールド） オプション	WEIGHT	[テーブルのフィールド：任意]	ポイントレイヤのフィールド。生成されるカウントは、そのポリゴンに含まれるポイントの重み属性の合計になります。重み属性が数値でない場合、カウントは 0 となります。
分類属性 オプション	CLASSFIELD	[テーブルのフィールド：任意]	ポイントは選択された分類属性に基づいて分類され、同じ分類属性値を持つ複数のポイントがポリゴン内に存在する場合は、そのうちの 1 つだけをカウントします。従って、最終的なポリゴン内の点の個数は、ポリゴン内にある相異なるクラスの個数となります。
カウント属性名	FIELD	[文字列] デフォルト：'NUMPOINTS'	ポイントの個数を格納するフィールドの名前
カウント（Count）	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力レイヤの指定。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
カウント（Count）	OUTPUT	[ベクタ：ポリゴン]	ポイントの個数の新しい列を含む属性テーブルを持つ結果レイヤ

Python コード

Algorithm ID: `native:countpointsinpolygon`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

DBSCAN クラスタリング

DBSCAN（Density-based spatial clustering of applications with noise）アルゴリズムの 2 次元版実装に基づいて、ポイント地物をクラスタリングします。

このアルゴリズムには、最小クラスタサイズと、クラスタ化されたポイント間で許容される最大距離の 2 つのパラメータが必要です。

参考:

ST-DBSCAN **クラスタリング**, *K* **平均クラスタリング**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	`INPUT`	[ベクタ：ポイント]	解析したいレイヤ
最小クラスタサイズ（minPts）	`MIN_SIZE`	[数値] デフォルト： 5	クラスタを構成する最小の地物数
クラスタ化された点の最大距離（eps）	`EPS`	[数値] デフォルト： 1.0	2 つの地物が同一クラスタとなることのない距離（eps）

次のページに続く

表 27.89 – 前のページからの続き

ラベル	名前	データ型	説明
クラスタ（Clusters）	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	クラスタ化の結果を出力するベクタレイヤを指定します。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
境界点をノイズとして扱う（DBSCAN*） オプション	DBSCAN*	[ブール値] デフォルト：False	チェックした場合、クラスタの境界上のポイント自体はクラスタ化されていないポイントとして扱われ、クラスタの内側のポイントのみがクラスタ化されるものとしてタグ付けされます。
クラスタを示す属性名	FIELD_NAME	[文字列] デフォルト：'CLUSTER_ID'	クラスタ番号を保存するフィールドの名前
クラスタサイズを示す属性名	SIZE_FIELD_NAME	[文字列] デフォルト：'CLUSTER_SIZE'	同じクラスタにある地物の数のフィールドの名前

出力

ラベル	名前	データ型	説明
クラスタ（Clusters）	OUTPUT	[ベクタ：ポイント]	そのポイントが属するクラスタを設定するフィールドを持つ、オリジナルの地物を含むベクタレイヤ
クラスタ数	NUM_CLUSTERS	[数値]	検出されたクラスタ数

Python コード

Algorithm ID: `native:dbscanclustering`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

距離行列（distance matrix）

ポイント地物について、同じレイヤ内または別のレイヤ内の最も近い地物までの距離を計算します。

デフォルトメニュー: **ベクタ　解析ツール**

参考:

属性の最近傍結合

パラメータ

ラベル	名前	データ型	説明
入力点のレイヤ	INPUT	[ベクタ：ポイント]	（ポイント **からの** ）距離行列を計算するポイントレイヤ
入力点の ID を示す属性（フィールド）	INPUT_FIELD	[テーブルのフィールド：任意]	入力レイヤの地物を一意に識別するために使用するフィールド。出力属性テーブルで使用します。
ターゲット点のレイヤ	TARGET	[ベクタ：ポイント]	（ポイント **への** ）最近傍点を検索するポイントレイヤ
ターゲット点の ID を示す属性（フィールド）	TARGET_FIELD	[テーブルのフィールド：任意]	ターゲットレイヤの地物を一意に識別するために使用するフィールド。出力属性テーブルで使用します。

次のページに続く

表 27.91 – 前のページからの続き

ラベル	名前	データ型	説明
出力形式	MATRIX_TYPE	[列挙型] デフォルト： 0	さまざまな計算タイプが利用できます： • 0 --- 線形距離行列（N * *k* x 3）：各入力点について、距離の近い *k* 個のターゲット点までの距離を返します。出力行列は入力点ごとに最大 *k* 行からなり、各行について 3 つの列：*InputID* 、*TargetID* 、*Distance* を持ちます。 • 1 --- 標準距離行列（N x T） • 2 --- 距離統計行列（平均、標準偏差、最小、最大）：各入力点について、ターゲット点に対する距離の統計量を返します。
計算する近傍点の個数（0 ならすべての点を計算）	NEAREST_POINTS	[数値] デフォルト： 0	距離を計算するポイントをターゲットレイヤ内のすべてのポイント（*0* ）とするか、最近傍の一定数（ *k* ）に制限するかを選択できます。
距離行列（distance matrix）	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力するベクタレイヤの指定。次のいずれかです: • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
距離行列（distance matrix）	OUTPUT	[ベクタ：ポイント]	各入力地物の距離計算を含むポイント（または「線形（N * *k* x 3）」の場合はマルチポイント）ベクタレイヤ。その地物と属性テーブルは、選択した出力行列の種類によって異なります。

Python コード

Algorithm ID: `qgis:distancematrix`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最寄りのハブの距離（ハブへの線）

入力ベクタの各地物をハブレイヤの最も近い地物に結合する線を作成します。距離は各地物の **中心** に基づいて計算されます。

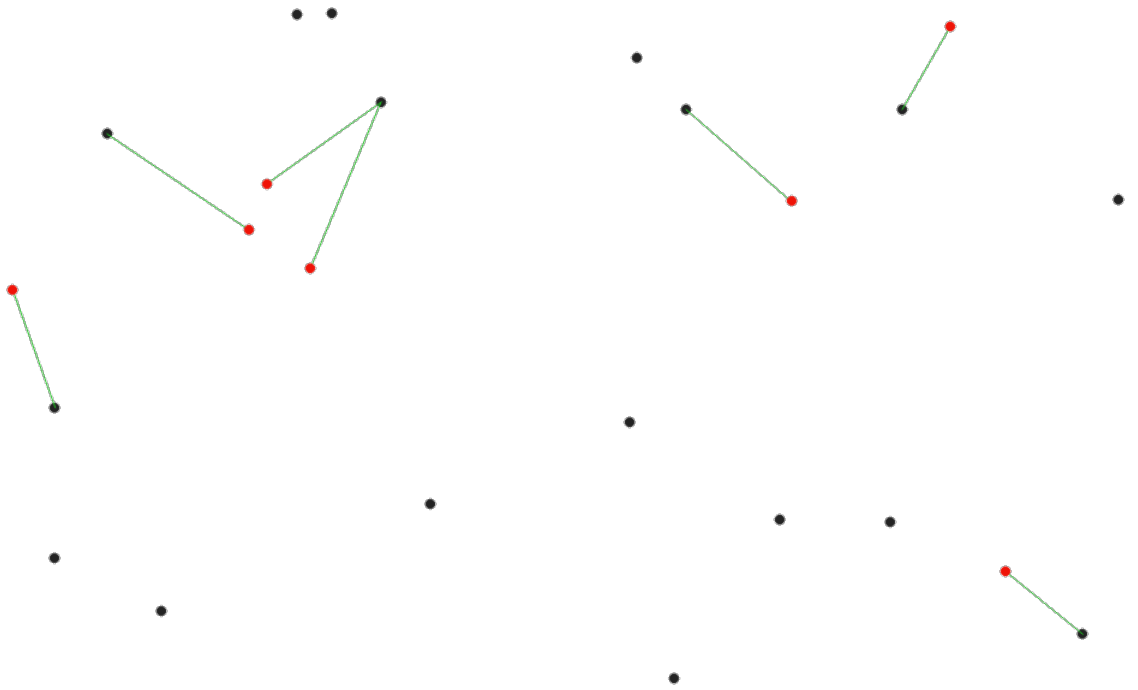


図 27.32: 赤色の入力地物に対する最近傍ハブの表示

参考:

最寄りのハブの距離 、 **属性の最近傍結合**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	最近傍の地物を探すベクタレイヤ
ハブレイヤ	HUBS	[ベクタ：任意]	検索対象の地物を含むベクタレイヤ
ハブレイヤからコピーされる属性（ハブの ID を想定）	FIELD	[テーブルのフィールド：任意]	ハブレイヤの地物を一意に識別するために使用するフィールド。出力属性テーブルで使用します。
計測単位	UNIT	[列挙型] デフォルト：0	最近傍の地物への距離の単位 • 0 --- メートル • 1 --- フィート • 2 --- マイル • 3 --- キロメートル • 4 --- レイヤの単位
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	合致した点を結んだ出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	入力地物の属性、最近傍の地物の ID、計算された距離の属性を持つラインベクタレイヤ

Python コード

Algorithm ID: qgis:distancetonearesthublinetohub

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最寄りのハブの距離

入力地物の **中心** を表すポイントレイヤを作成します。これには、（中心点に基づく）最も近い地物の ID と、ポイント間の距離の 2 つのフィールドが追加されています。

参考:

最寄りのハブの距離（ハブへの線） 、 **属性の最近傍結合**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	最近傍の地物を探すベクタレイヤ
ハブレイヤ	HUBS	[ベクタ：任意]	検索対象の地物を含むベクタレイヤ
ハブレイヤからコピーされる属性（ハブの ID を想定）	FIELD	[テーブルのフィールド：任意]	ハブレイヤの地物を一意に識別するために使用するフィールド。出力属性テーブルで使用します。
計測単位	UNIT	[列挙型] デフォルト： 0	最近傍の地物への距離の単位 • 0 --- メートル • 1 --- フィート • 2 --- マイル • 3 --- キロメートル • 4 --- レイヤの単位
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	最寄りのハブを持った出力ポイントベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	ソース地物の属性、最近傍の地物の ID、計算された距離を持った、ソース地物の中心を表すポイントベクタレイヤ

Python コード

Algorithm ID: `qgis:distancetonearesthubpoints`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ハブ＆スポーク図

スポークレイヤのポイントから対応するハブレイヤのポイントへ線をつなぐことで、ハブ＆スポーク図を作成します。

どのハブが各ポイントに対応するかは、ハブポイント上のハブ ID フィールドとスポークポイント上のスポーク ID フィールドの一致に基づいて決定されます。

入力レイヤがポイントレイヤでない場合には、ジオメトリの内部保証点（point on surface）を接続する地点として使用します。

オプションとして、測地線（回転楕円体上の最短経路）を作成することもできます。測地線モードを使用する場合、作成する線分を日付変更線（経度 ±180 度）で分割することもできます。これにより、線分のレンダリングが向上します。また、頂点間の距離も指定できます。距離が短いほど線分は密になり、正確な線になります。

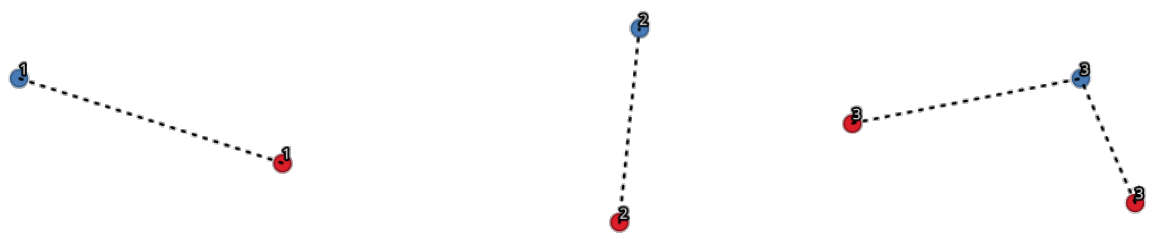


図 27.33: 共通のフィールド / 属性に基づくポイントの連結

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ハブレイヤ	HUBS	[ベクタ：任意]	入力レイヤ
ハブ ID 属性	HUB_FIELD	[テーブルのフィールド：任意]	連結に使用するハブレイヤの ID フィールド
コピーするハブレイヤの属性（すべての属性をコピーする場合は空のまま） オプション	HUB_FIELDS	[テーブルのフィールド：任意] [リスト]	コピーするハブレイヤのフィールド（複数可）。フィールドが何も選択されない場合は、すべてのフィールドをコピーします。
スポークレイヤ	SPOKES	[ベクタ：任意]	スポーク点のレイヤ
スポーク ID 属性	SPOKE_FIELD	[テーブルのフィールド：任意]	連結に使用するスポークレイヤの ID フィールド
コピーするスポークレイヤの属性（すべての属性をコピーする場合は空のまま） オプション	SPOKE_FIELDS	[テーブルのフィールド：任意] [リスト]	コピーするスポークレイヤのフィールド（複数可）。フィールドが何も選択されない場合は、すべてのフィールドをコピーします。
測地線を作成	GEODESIC	[ブール値] デフォルト：False	測地線（回転楕円体上の最短経路）を作成します
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ハブラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
頂点間の距離（測地線のみ）	GEODESIC_DISTANC	[数値] デフォルト：1000.0（キロメートル）	連続する頂点間の距離（キロメートル単位）。距離が小さいほど線分は密になり、正確な線となる

次のページに続く

表 27.94 – 前のページからの続き

ラベル	名前	データ型	説明
日付変更線（経度 180 度線）で線を切断	ANTIMERIDIAN_SPL	[ブール値] デフォルト：False	経度 ±180 度で線を切断する（ラインのレンダリングが崩れないようにするため）

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	入力レイヤの適合した点を接続した、結果ラインレイヤ

Python コード

Algorithm ID: native:hublines

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

K 平均クラスタリング

各入力地物について、2 次元的な距離に基づく K 平均法によるクラスタ番号を計算します。

K平均クラスタリングは、地物を k 個のクラスタに分割し、各地物が最も近い平均を持つクラスタに属するようにすることを目的とします。平均点は、クラスタ化された地物の重心によって表されます。

入力ジオメトリがラインやポリゴンの場合には、クラスタリングは地物の重心に基づいて行われます。

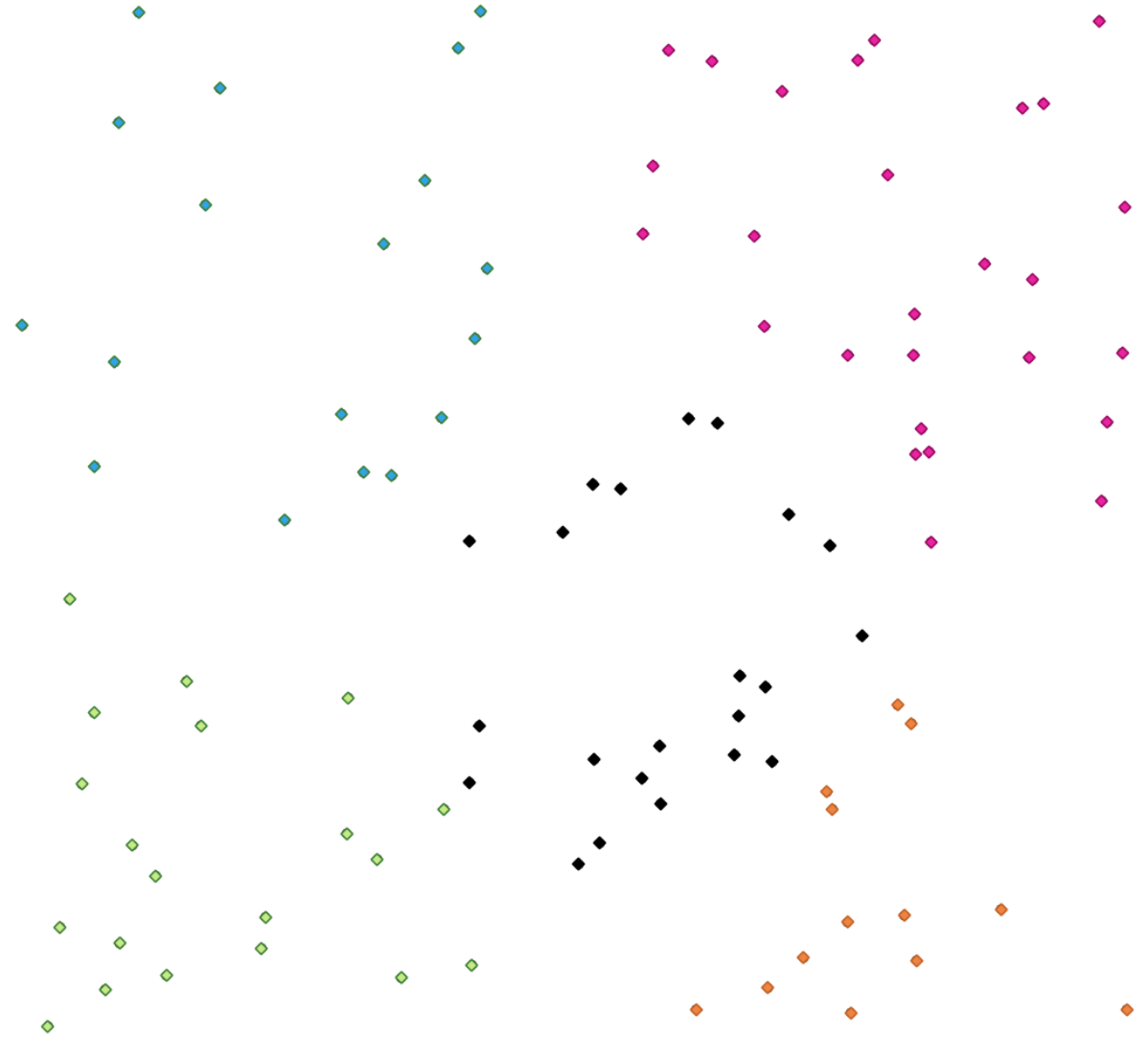


図 27.34: 5 クラスのポイントクラスタ

参考:

DBSCAN **クラスタリング**, *ST-DBSCAN* **クラスタリング**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	解析したいレイヤ
クラスタ数	CLUSTERS	[数値] デフォルト：5	作成したい地物クラスタ数
クラスタ（Clusters）	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	生成されたクラスタの出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
クラスタを示す属性名	FIELD_NAME	[文字列] デフォルト：'CLUSTER_ID'	クラスタ番号を保存するフィールドの名前
クラスタサイズを示す属性名	SIZE_FIELD_NAME	[文字列] デフォルト: 'CLUSTER_SIZE'	同じクラスタにある地物の数のフィールドの名前

出力

ラベル	名前	データ型	説明
クラスタ（Clusters）	OUTPUT	[ベクタ：任意]	地物が属するクラスタとその番号を指定するフィールドを持つ、オリジナルの地物を格納しているベクタレイヤ

Python コード

Algorithm ID: `native:kmeansclustering`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ユニーク値のリスト

属性テーブルフィールドのユニーク値をリストし、その数を数えます。

デフォルトメニュー: **ベクタ　解析ツール**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	解析したいレイヤ
対象属性（フィールド）	FIELDS	[テーブルのフィールド：任意]	分析したいフィールド
出力レイヤ オプション	OUTPUT	[テーブル] デフォルト：[一時レイヤを作成]	ユニーク値のサマリテーブルレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
HTML レポートの出力 オプション	OUTPUT_HTML_FILE	[html] デフォルト：[一時ファイルに保存]	**プロセシング　結果ビューア** のユニーク値の HTML レポート。次のいずれかです： • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[テーブル]	ユニーク値の集計テーブルレイヤ
HTML レポートの出力	OUTPUT_HTML_FILE	[html]	ユニーク値の HTML レポート。 **プロセシング　結果ビューア** からアクセスできます。
ユニークな値の総数	TOTAL_VALUES	[数値]	入力フィールドにあるユニーク値の数
連結したユニーク値	UNIQUE_VALUES	[文字列]	入力フィールドで見つかったユニーク値のリストをコンマ区切りで繋げた文字列

Python コード

Algorithm ID: `qgis:listuniquevalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

加重平均座標（重心の平均）

入力レイヤのジオメトリの重心を計算したポイントレイヤを作成します。

重心を計算する際に、各地物に適用される重み付けの属性を指定できます。

パラメータで属性を選択すると、地物はこのフィールドの値でグループ化されます。出力レイヤには、レイヤ全体の重心を示す単一の点ではなく、各カテゴリの地物の重心が含まれるようになります。

デフォルトメニュー: **ベクタ　解析ツール**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
重み属性（フィールド） オプション	WEIGHT	[テーブルのフィールド：数値]	重み付き平均を計算したい場合に使用するフィールド
ユニーク ID 属性	UID	[テーブルのフィールド：数値]	グループ化した平均の計算を行う上でのユニークフィールド
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	結果の（ポイントベクタ）レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	結果のポイントレイヤ

Python コード

Algorithm ID: native:meancoordinates

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

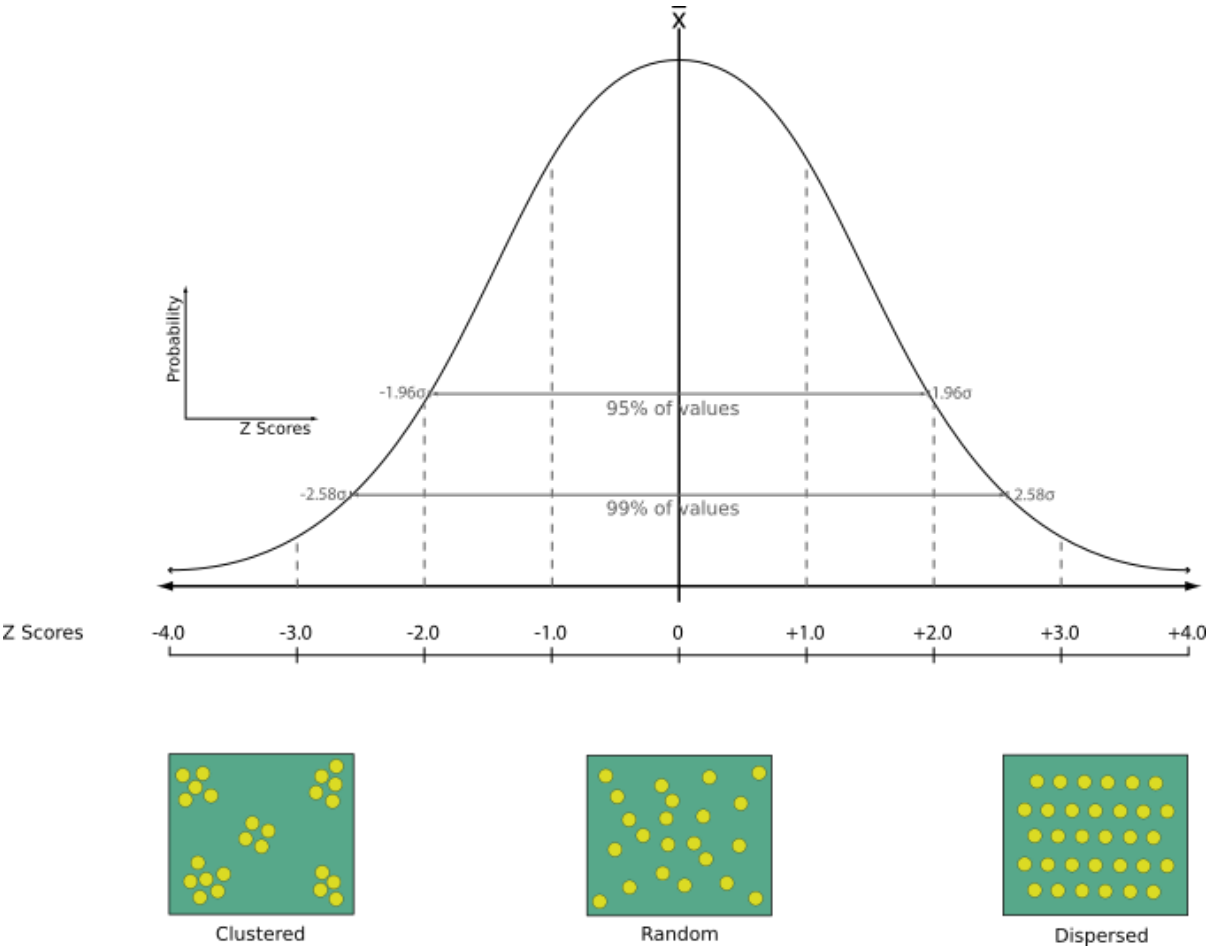
algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最近傍解析

ポイントレイヤの最近傍解析を実行します。出力結果は、データがどのように分布しているか（クラスター化しているか、ランダムに分布しているか等）を表します。

出力結果は、計算された以下の統計値を含む HTML ファイルとして生成されます：

- 観測平均距離
- 推定平均距離
- 最近傍インデックス
- 頂点数
- Z-スコア：Z-スコアを正規分布と比較すると、データがどのように分布しているかがわかります。Z-スコアが低いと、データは空間的にランダムなプロセスの結果である可能性が低いことを意味し、反対に Z-スコアが高いと、データは空間的にランダムなプロセスの結果である可能性が高いことを意味します。



デフォルトメニュー: **ベクタ　解析ツール**

参考:

属性の最近傍結合

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	統計値を計算したいポイントベクタレイヤ
最 近 傍（Nearest neighbor） オプション	OUTPUT_HTML_FILE	[html] デフォルト：[一時ファイルに保存]	計算した統計を出力する HTML ファイルの指定。次のいずれかです: • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
最 近 傍（Nearest neighbor）	OUTPUT_HTML_FILE	[html]	統計量の計算結果の HTML ファイル
観測平均距離	OBSERVED_MD	[数値]	観測平均距離
推定平均距離	EXPECTED_MD	[数値]	推定平均距離
最近傍インデックス	NN_INDEX	[数値]	最近傍インデックス
点の数	POINT_COUNT	[数値]	頂点数
Z-スコア	Z_SCORE	[数値]	Z-スコア

Python コード

Algorithm ID: `native:nearestneighbouranalysis`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

重なり分析

入力レイヤの地物が選択したオーバーレイレイヤの地物に対して重なる面積と割合を計算します。

入力地物が選択された各オーバーレイレイヤと重なる部分の面積と割合が、新しい属性として出力レイヤに追加されます。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力レイヤ
オーバーレイレイヤ	LAYERS	[ベクタ：任意] [リスト]	オーバーレイレイヤ
重なり	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
グリッドサイズ NEW in 3.28 オプション	GRID_SIZE	[数値] デフォルト：未設定	指定された場合、入力ジオメトリは指定されたサイズのグリッドにスナップされ、結果の頂点は同じグリッド上で計算されます。GEOS 3.9.0 以上が必要です。

出力

ラベル	名前	データ型	説明
重なり	OUTPUT	[入力レイヤと同じ]	入力地物と選択された各オーバーレイレイヤとの重なり（地図単位の面積およびパーセンテージ）のフィールドが追加された出力レイヤ

Python コード

Algorithm ID: `native:calculatevectoroverlaps`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地物間の最短線

`NEW in 3.24`

起点レイヤと出力レイヤ間の最短線としてラインレイヤを作成します。デフォルトでは、出力レイヤの最初の最近傍地物のみが考慮されます。n 番目に近い地物の数を指定できます。最大距離を指定すると、その距離より近い地物のみが考慮されます。

出力地物は、起点レイヤの全ての属性、n 最近傍地物の全ての属性、及び距離の追加フィールドを格納します。

重要: このアルゴリズムでは、距離については純粋にデカルト計算を使用し、地物の近接度を決定する際に測地線や楕円体の特性は考慮しません。測定及び出力の座標系は、起点レイヤの座標系に基づいてい ます。

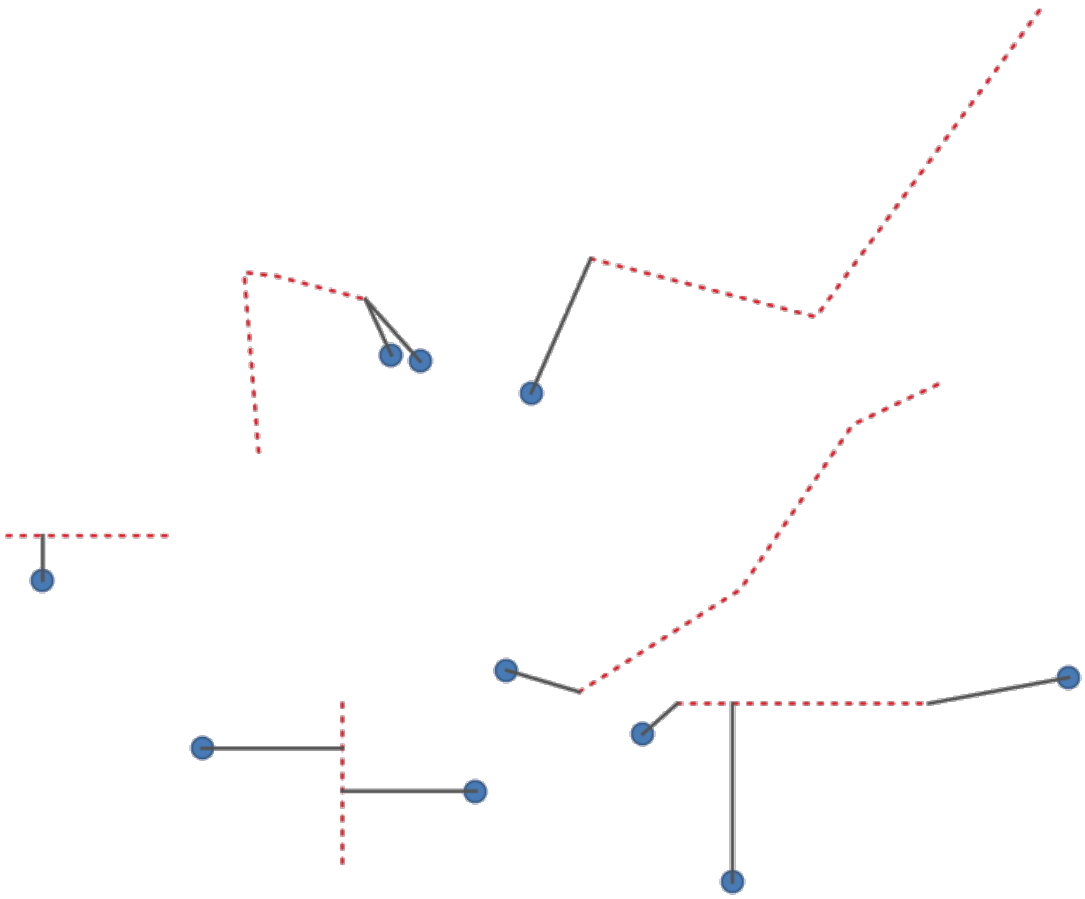


図 27.35: ポイント地物からラインへの最短線

パラメータ

ラベル	名前	データ型	説明
起点レイヤ	SOURCE	[ベクタ：任意]	その最近傍を検索する起点レイヤ
出力レイヤ	DESTINATION	[ベクタ：任意]	その中にある最近傍を検索する対象レイヤ
方法	METHOD	[列挙型] デフォルト：0	最短距離の計算法。可能な値は次のとおりです： • 0 -- 地物の最も近い点までの距離 • 1 -- 地物の重心までの距離
考慮する隣接地物の数	NEIGHBORS	[数値] デフォルト：1	探索する近隣地物の最大数
最大距離 オプション	DISTANCE	[数値]	この距離より近い地物だけが考慮されます。
最短線	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	起点地物と出力レイヤの最近傍地物を結ぶラインベクタレイヤ。起点地物と目標地物の全ての属性と、計算された距離が含まれます。

Python コード

Algorithm ID: native:shortestline

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ST-DBSCAN クラスタリング

Density-based clustering of applications with noise (ST-DBSCAN) アルゴリズムの 2 次元実装に基づくポイント地物のクラスタリング。

参考:

DBSCAN **クラスタリング**, *K* **平均クラスタリング**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	解析したいレイヤ
日付・時間フィールド	DATETIME_FIELD	[tablefield: 日付]	時間情報を格納しているフィールド
最小クラスタサイズ（minPts）	MIN_SIZE	[数値] デフォルト： 5	クラスタを構成する最小の地物数
クラスタ化された点の最大距離（eps）	EPS	[数値] デフォルト： 1.0	2 つの地物が同一クラスタとなることのない距離（eps）
クラスタ化された点の最大継続時間	EPS2	[数値] デフォルト： 0.0（日）	2 つの地物が同じクラスターに属することができない時間 (eps2)。利用可能な時間単位はミリ秒、秒、分、時、日、週です。
クラスタ（Clusters）	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	クラスタ化の結果を出力するベクタレイヤを指定します。次のいずれかです: • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
境界点をノイズとして扱う（DBSCAN*） オプション	DBSCAN*	[ブール値] デフォルト：False	チェックした場合、クラスタの境界上のポイント自体はクラスタ化されていないポイントとして扱われ、クラスタの内側のポイントのみがクラスタ化されるものとしてタグ付けされます。
クラスタを示す属性名	FIELD_NAME	[文字列] デフォルト：'CLUSTER_ID'	クラスタ番号を保存するフィールドの名前
クラスタサイズを示す属性名	SIZE_FIELD_NAME	[文字列] デフォルト: 'CLUSTER_SIZE'	同じクラスタにある地物の数のフィールドの名前

出力

ラベル	名前	データ型	説明
クラスタ（Clusters）	OUTPUT	[ベクタ：ポイント]	そのポイントが属するクラスタを設定するフィールドを持つ、オリジナルの地物を含むベクタレイヤ
クラスタ数	NUM_CLUSTERS	[数値]	検出されたクラスタ数

Python コード

Algorithm ID: `native:stdbscanclustering`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

出力レイヤ

親クラスに応じたフィールドの統計量を計算します。親クラスは、他のフィールドの値の組み合わせです。

パラメータ

ラベル	名前	データ型	説明
入力ベクタレイヤ	INPUT	[ベクタ：任意]	ユニークなカテゴリと値を持つ入力ベクタレイヤ
集計する属性（空の場合はカウントのみ） オプション	VALUES_FIELD_NAM	[テーブルのフィールド：任意]	空の場合はカウントのみ計算します
カテゴリ分けする属性	CATEGORIES_FIELD	[ベクタ：任意] [リスト]	カテゴリを定義するフィールド（組み合わせ可）
カテゴリ別の統計量	OUTPUT	[テーブル] デフォルト：[一時レイヤを作成]	生成した統計量の出力テーブルを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
カテゴリ別の統計量	OUTPUT	[テーブル]	統計量のテーブル

集計する属性の型に応じて、各カテゴリについて以下の統計量が返されます：

統計	文字列	数値	日付
個数（ COUNT ）	☑	☑	☑
ユニーク値（ UNIQUE ）	☑		☑
空値（null）（ EMPTY ）	☑		☑
非 null 値（ FILLED ）	☑		☑
最小値（ MIN ）	☑	☑	☑

次のページに続く

表 27.99 – 前のページからの続き

統計	文字列	数値	日付
最大値（MAX）	☑	☑	☑
範囲（RANGE）		☑	
合計（SUM）		☑	
平均（MEAN）		☑	
中央値（MEDIAN）		☑	
標準偏差（STD_DEV）		☑	
分散係数（CV）		☑	
最稀値（最も頻度の少ない値 - MINORITY）		☑	
最頻値（最も頻度の多い値 - MAJORITY）		☑	
第１四分位（FIRSTQUARTILE）		☑	
第３四分位（THIRDQUARTILE）		☑	
四分位範囲（IQR）		☑	
最小長さ（MIN_LENGTH）	☑		
平均の長さ（MEAN_LENGTH）	☑		
最大長さ（MAX_LENGTH）	☑		

Python コード

Algorithm ID: `qgis:statisticsbycategories`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線長の合計

ポリゴンレイヤとラインレイヤを使用して、各ポリゴンを横切るラインの合計長と合計数を計測します。

結果として得られるレイヤは、入力ポリゴンレイヤと同じ地物ですが、各ポリゴンを横切るラインの長さと本数の 2 つ属性が追加されています。

☑ ポリゴン地物の **地物の *In-place* 編集** が可能です

デフォルトメニュー: **ベクタ　解析ツール**

パラメータ

ラベル	名前	データ型	説明
線レイヤ	LINES	[ベクタ：ライン]	入力ベクタラインレイヤ
ポリゴン（Polygons）	POLYGONS	[ベクタ：ポリゴン]	ポリゴンベクタレイヤ
交差する線の総延長を格納するフィールドの名前	LEN_FIELD	[文字列] デフォルト：'LENGTH'	線の長さのフィールド名
交差する線の数を格納するフィールドの名前	COUNT_FIELD	[文字列] デフォルト：'COUNT'	線の数のフィールド名
ポリゴンと交差する線の総延長	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	生成した統計量を持った出力ポリゴンレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ポリゴンと交差する線の総延長	OUTPUT	[ベクタ：ポリゴン]	ライン長とライン数のフィールドを持つ出力ポリゴンレイヤ

Python コード

Algorithm ID: native:sumlinelengths

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.16 ベクタ作成

オフセット線の配列

レイヤ内のライン地物をオフセットしたものを複数作成することによって、ライン地物のコピーを作成します。新しいラインはそれぞれ、指定された距離ずつオフセットされます。

正の値の距離はラインを左にオフセットし、負の値の距離は右にオフセットします。

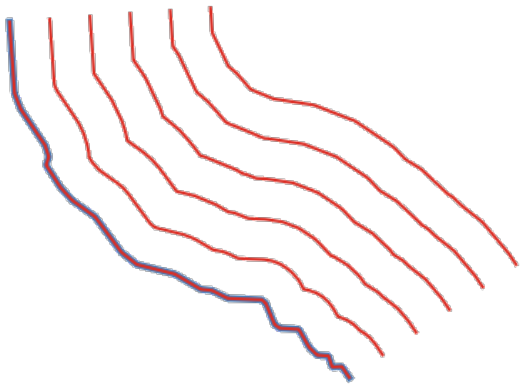


図 27.36: 青色ラインはソースレイヤ、赤色ラインはオフセットされたもの

ライン地物の **地物の** *In-place* **編集** が可能です。

参考:

オフセット線 、 **平行移動した地物の配列**

パラメータ

基本パラメータ

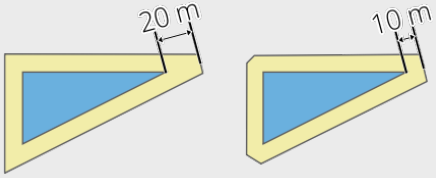
ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	オフセットの作成に使用する入力ライン ベクタレイヤ
コピーの数	COUNT	[数値] デフォルト： 10	各地物について作成したいオフセットのコピーの数
ステップ距離	OFFSET	[数値] デフォルト： 1.0	2 つの連続するオフセットのコピー間の距離

次のページに続く

表 27.101 – 前のページからの続き

ラベル	名前	データ型	説明
オフセット線	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	オフセット地物の出力ラインレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
セグメント	SEGMENTS	[数値] デフォルト：8	丸みを帯びたオフセットの四分円を近似するために使用するセグメントの数
継ぎ目スタイル	JOIN_STYLE	[列挙型] デフォルト：0	ラインの角をオフセットするときの継ぎ目スタイルの指定。次のいずれかです： • 0 --- Round • 1 --- Miter • 2 --- Bevel  図 27.37: round、miter、bevel の継ぎ目スタイル
miter 制限	MITER_LIMIT	[数値] デフォルト：2.0	miter 継ぎ目を作成する際に使用するオフセットジオメトリからの最大距離を、オフセット距離の係数として設定します（miter 継ぎ目スタイルにのみ適用されます）。最小値: 1.0  図 27.38: 10m バッファで制限 2 の場合と、10m バッファで制限 1 の場合

出力

ラベル	名前	データ型	説明
オフセット線	OUTPUT	[ベクタ：ライン]	オフセット地物の出力ラインレイヤ。元の地物もコピーされます。

Python コード

Algorithm ID: native:arrayoffsetlines

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

平行移動した地物の配列

入力レイヤの地物を平行移動させたコピーを複数作成します。各コピーは地物を X、Y、Z 軸方向に指定間隔で段階的に移動させたものです。

ジオメトリに存在する M 値も変換できます。

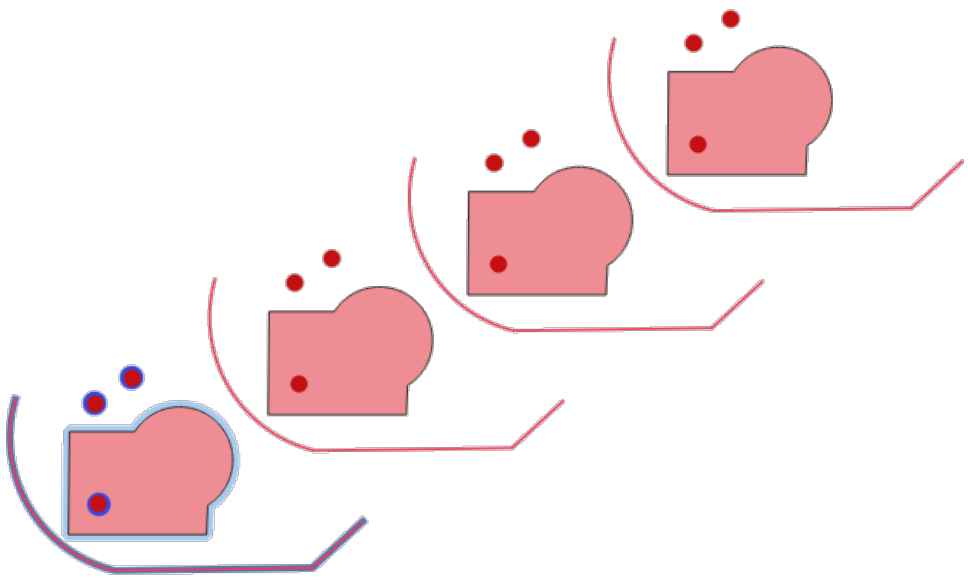


図 27.39: 入力レイヤは青色、平行移動した地物の出力レイヤは赤色

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

平行移動（*translate*）、**オフセット線の配列**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	平行移動させたい入力ベクタレイヤ
コピーの数	COUNT	[数値] デフォルト： 10	各地物について作成したいコピーの数
ステップ距離（X 軸）	DELTA_X	[数値] デフォルト： 0.0	X 軸方向に適用する移動量
ステップ距離（Y 軸）	DELTA_Y	[数値] デフォルト： 0.0	Y 軸方向に適用する移動量
ステップ距離（Z 軸）	DELTA_Z	[数値] デフォルト： 0.0	Z 軸方向に適用する移動量
ステップ距離（M 値）	DELTA_M	[数値] デフォルト： 0.0	M 値に適用する移動量
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	平行移動させた地物の出力ベクタレイヤ。元の地物もコピーされます。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	平行移動させた地物の出力ベクタレイヤ。元の地物もコピーされます。

Python コード

Algorithm ID: `native:arraytranslatedfeatures`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッドを作成

指定した範囲を覆うグリッドのベクタレイヤを作成します。グリッドのセルは以下のように様々な形状があります：

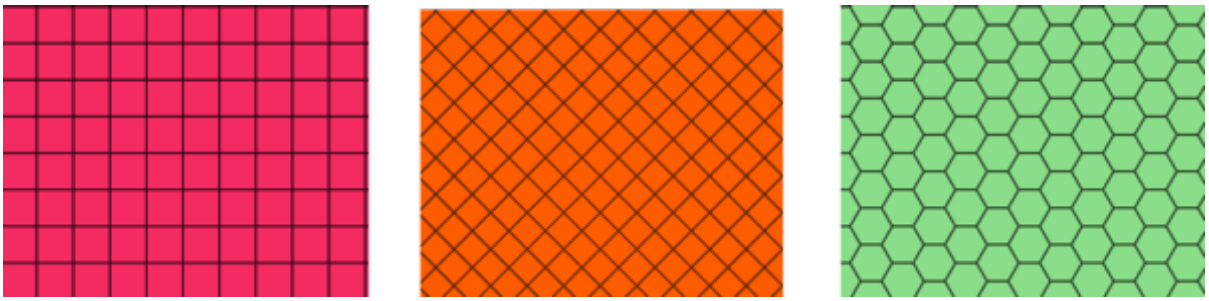


図 27.40: 様々なグリッドセル形状

グリッドの各要素のサイズは、水平方向・垂直方向の間隔を使用して定義されます。

出力レイヤの CRS を指定する必要があります。

グリッドの範囲と間隔の値は、この CRS の座標と長さ単位で表されている必要があります。

デフォルトメニュー ： **ベクタ**　**調査ツール**

パラメータ

ラベル	名前	データ型	説明
グリッドタイプ	TYPE	[列挙型] デフォルト： 0	グリッドの形状。次のいずれかです： • 0 --- 点（Point） • 1 --- ライン • 2 --- 長方形（Polygon） • 3 --- 菱形（Polygon） • 4 --- 六角形（Polygon）
グリッドの範囲	EXTENT	[範囲]	グリッドの範囲 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
水平方向の間隔	HSPACING	[数値] デフォルト： 1.0	グリッドのセルの X 軸方向サイズ
垂直方向の間隔	VSPACING	[数値] デフォルト： 1.0	グリッドのセルの Y 軸方向サイズ
水平方向の重なり	HOVERLAY	[数値] デフォルト： 0.0	2 つの連続するグリッドセルの X 軸方向の重なり距離
垂直方向の重なり	VOVERLAY	[数値] デフォルト： 0.0	2 つの連続するグリッドセルの Y 軸方向の重なり距離
出力グリッドの CRS	CRS	[crs] デフォルト： **プロジェクト** *CRS*	グリッドに適用する座標参照系

次のページに続く

表 27.104 – 前のページからの続き

ラベル	名前	データ型	説明
グリッド	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	結果のグリッドベクタレイヤ。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
グリッド	OUTPUT	[ベクタ：任意]	結果のグリッドベクタレイヤ。出力のジオメトリタイプ（ポイント、ライン、ポリゴン）は **グリッドタイプ** によります。

Python コード

Algorithm ID: `native:creategrid`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

テーブルから点レイヤを作成

座標フィールドを含むカラムを持つテーブルから、ポイントレイヤを作成します。

X、Y 座標に加えて、Z 座標や M 値のフィールドも指定できます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤまたはテーブル
X 属性	XFIELD	[テーブルのフィールド：任意]	X 座標のフィールド
Y 属性	YFIELD	[テーブルのフィールド：任意]	Y 座標のフィールド
Z 属性 オプション	ZFIELD	[テーブルのフィールド：任意]	Z 座標のフィールド
M 値の属性 オプション	MFIELD	[テーブルのフィールド：任意]	M 値のフィールド
ラスタの CRS	TARGET_CRS	[crs] デフォルト：EPSG:4326	レイヤに使用する座標参照系。指定した座標値は、この座標参照系に対応したものであることが想定されます。
テーブルによる点	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	結果のポイントレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
テーブルによる点	OUTPUT	[ベクタ：ポイント]	結果のポイントレイヤ

Python コード

Algorithm ID: native:createpointslayerfromtable

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線に沿った点群の生成

入力のラスタレイヤとラインレイヤから、ポイントベクタレイヤを作成します。

ラインレイヤと交わるピクセルの中心に対応したポイントです。

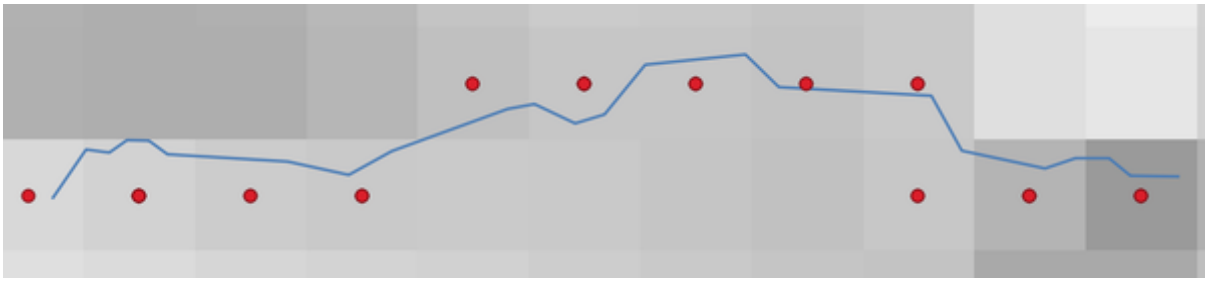


図 27.41: ピクセル中心のポイント

パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
入力ベクタ	INPUT_VECTOR	[ベクタ：ライン]	入力ラインベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	結果のピクセル中心のポイントレイヤ。 次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	結果のピクセル中心のポイントレイヤ

Python コード

Algorithm ID: `qgis:generatepointspixelcentroidsalongline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ポリゴン内の点を作成

入力のラスタレイヤとポリゴンレイヤから、ポイントベクタレイヤを作成します。

ポリゴンレイヤと交わるピクセルの中心に対応したポイントです。

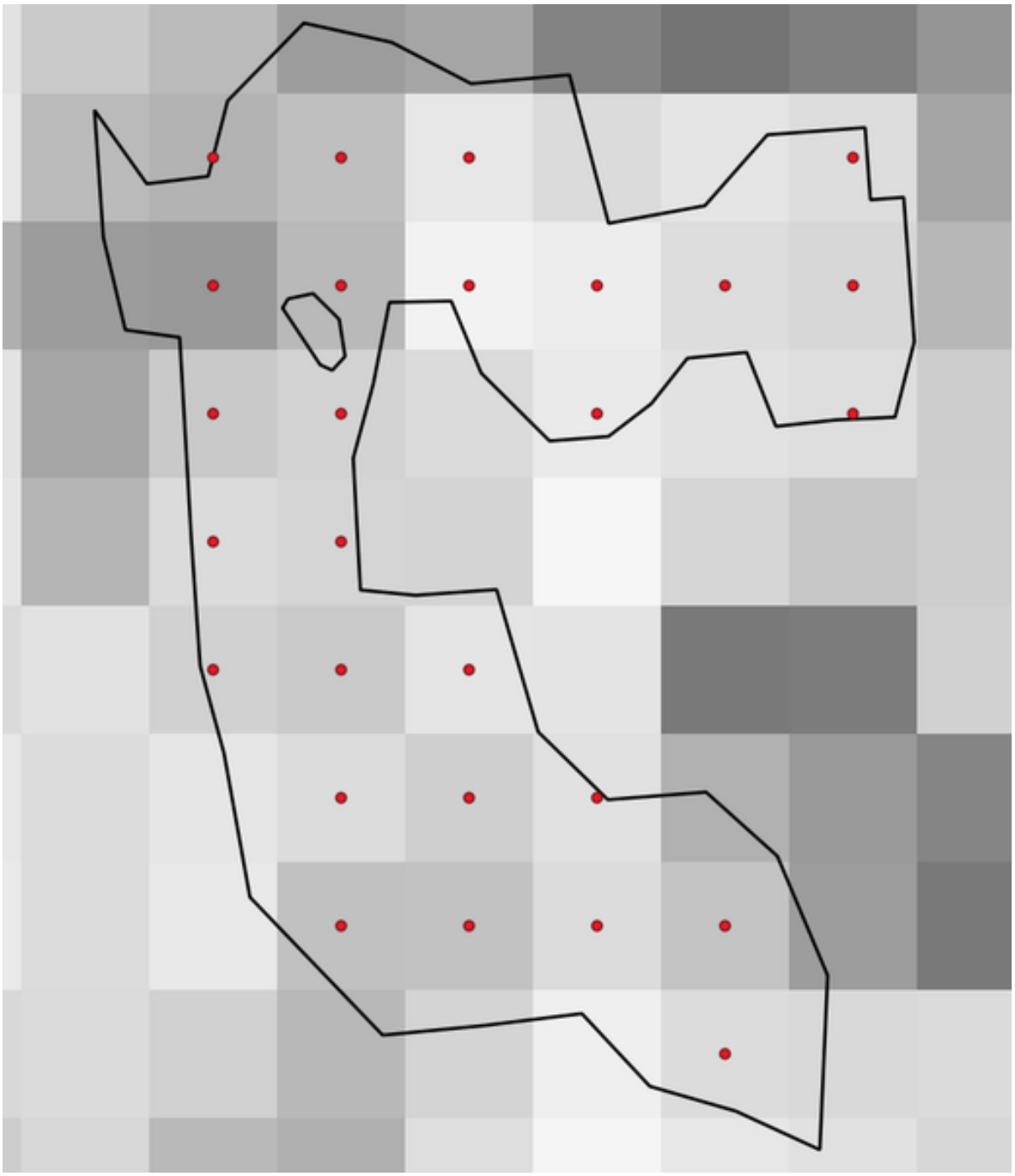


図 27.42: ピクセル中心のポイント

パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
入力ベクタ	INPUT_VECTOR	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
ピクセル重心	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	結果のピクセル中心のポイントレイヤ。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ピクセル重心	OUTPUT	[ベクタ：ポイント]	結果のピクセル中心のポイントレイヤ

Python コード

Algorithm ID: `native:generatepointspixelcentroidsinsidepolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオタグ（位置情報）付きの写真

ソースフォルダ内の JPEG 画像から、ジオタグの位置に対応したポイントレイヤを作成します。

ポイントレイヤには、ジオタグを読み込むことができる入力ファイルごとに 1 つの PointZ 地物が作られます。ジオタグからの高度情報は、ポイントの Z 値を設定するために使用されます。

経度と緯度の他に、高度、方向、タイムスタンプの情報も写真に含まれていれば、属性としてポイントに追加されます。

パラメータ

ラベル	名前	データ型	説明
写真のあるフォルダ	FOLDER	[フォルダ]	ジオタグ付き写真が含まれるソースフォルダのパス
下位フォルダも検索	RECURSIVE	[ブール値] デフォルト：False	チェックされた場合、フォルダ内のフォルダも検索します
出力レイヤ オプション	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ジオタグ付き写真のポイントベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
写真のテーブルが無効です オプション	INVALID	[テーブル] デフォルト：[出力をスキップ]	読み込めない、またはジオタグが付いていない写真の結果テーブルを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	ジオタグ付き写真のポイントベクタレイヤ。レイヤの属性フォームには自動的にパスが入力され、写真のプレビュー設定が行われます。
写真のテーブルが無効です オプション	INVALID	[テーブル]	読み込めない、またはジオタグが付いていない写真のテーブルを作成できます

Python コード

Algorithm ID: `native:importphotos`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

点を線に変換

入力ポイントレイヤの式またはフィールドで定義された順序で点を結合することにより、ポイントレイヤをラインレイヤに変換します。

ライン地物を区別するため、ポイントをフィールドまたは式でグループ化できます。

ラインベクタレイヤに加えて、結果のラインを説明するテキストファイルの出力も得られます。テキストファイルには、開始点、（方位角に対する）方位/方向、距離の並びのデータが含まれます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
閉じたパスを作る	CLOSE_PATH	[ブール値] デフォルト：False	チェックされた場合、最初と最後のポイントをつないで、閉じた経路を作成します

次のページに続く

表 27.107 – 前のページからの続き

ラベル	名前	データ型	説明
式による並べ替え オプション	ORDER_EXPRESSION	[式]	パス内のポイントの接続順序を指定するフィールドまたは式。設定されていない場合は、地物 ID (`$id`) が使われます。
文字列の自然なソート オプション	NATURAL_SORT	[ブール値] デフォルト：False	チェックされた場合、地物は指定された式に基づいて自然にソートされる（すなわち、'a9' < 'a10'）。
グループを示す式 オプション	GROUP_EXPRESSION	[式]	フィールドまたは式で同じ値を持つポイント地物は、同じラインにグループ化されます。設定されていない場合は、すべての入力ポイントで 1 つのパスが描かれます。
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	パスのラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
テキスト出力先ディレクトリ オプション	OUTPUT_TEXT_DIR	[フォルダ] デフォルト：[出力をスキップ]	ポイントとパスの説明ファイルの出力先ディレクトリを指定します。次のいずれかです： • 出力をスキップ • 一時ディレクトリに保存 • ディレクトリに保存します

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	パスのラインベクタレイヤ
テキスト出力先ディレクトリ	OUTPUT_TEXT_DIR	[フォルダ]	ポイントとパスの説明ファイルの出力先ディレクトリ

Python コード

Algorithm ID: `native:pointstopath`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線に沿ったランダム点群

ラインレイヤの線上にポイントが乗っているポイントレイヤを新たに作成します。

入力レイヤ内の各ラインについて、指定した数のポイントが結果レイヤに追加されます。ポイントを追加する手続きは、以下のように行われます：

1. 入力レイヤからランダムにライン地物を選択する
2. その地物がマルチパート地物の場合には、そのパートを一つランダムに選択する
3. そのラインのセグメントをランダムに選択する
4. そのセグメント上で位置をランダムに選択する

この手続きでは、ラインの曲線部分（比較的短いセグメント群）の方が直線部分（比較的長いセグメント）よりもポイントが多く発生しがちです。 **線に沿ったランダム点群（*randompointsalongline*）** アルゴリズムの出力は、 **線に沿ったランダム点群（*randompointsonlines*）** アルゴリズム（ラインに沿って平均的に均等にポイントを生成する）と比較して、曲線部分にポイントが多いことが以下の図からもわかります。

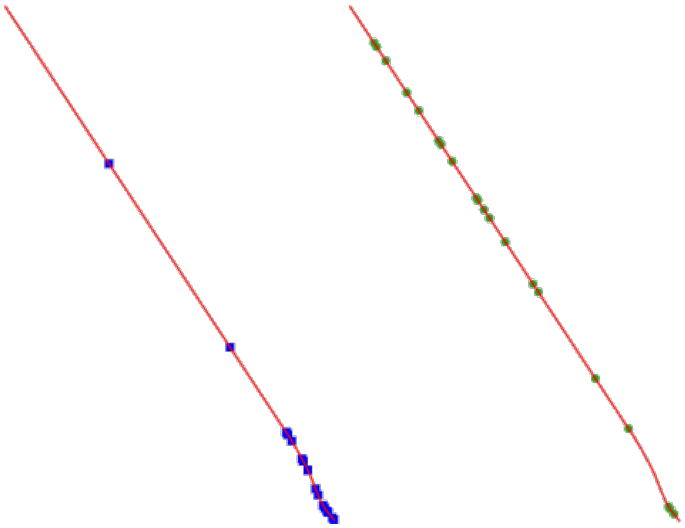


図 27.43: アルゴリズムの出力例。左： **線に沿ったランダム点群（本アルゴリズム、*randompointsalongline*）** 、右： **線に沿ったランダム点群（*randompointsonlines*）**

ポイントが互いに近くなりすぎることを避けるため、最小距離を指定できます。

参考:

線に沿ったランダム点群

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
点の数	POINTS_NUMBER	[数値] デフォルト： 1	作成したいポイントの数
点間距離の最小値	MIN_DISTANCE	[数値] デフォルト： 0.0	ポイント間の最小距離
ランダム点群出力	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ランダム点群の出力。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ランダム点群出力	OUTPUT	[ベクタ：ポイント]	ランダムなポイントの出力レイヤ

Python コード

Algorithm ID: `qgis:qgisrandompointsalongline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ランダム点群

指定された個数のランダムなポイントを指定された範囲内にすべて収まるように生成したポイントレイヤを新たに作成します。

最小距離を指定することで、ポイントが互いに近くなりすぎないようにすることができます。点間距離の最小値のために新たなポイントが作成できなくなる場合には、最小距離を小さくするか、最大試行回数を大きくしてください。

デフォルトメニュー ： ベクタ　 調査ツール

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
作成範囲	EXTENT	[範囲]	ランダム点群を生成するマップ範囲 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
点の数	POINTS_NUMBER	[数値] デフォルト： 1	作成したいポイント数
点間距離の最小値	MIN_DISTANCE	[数値] デフォルト： 0.0	ポイント間の最小距離
ラスタの CRS	TARGET_CRS	[crs] デフォルト： **プロジェクト** *CRS*	ランダム点群レイヤの CRS
ランダム点群出力	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ランダム点群の出力。次のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
最大試行回数	MAX_ATTEMPTS	[数値] デフォルト： 200	ポイントを配置するための最大試行回数

出力

ラベル	名前	データ型	説明
ランダム点群出力	OUTPUT	[ベクタ：ポイント]	ランダムなポイントの出力レイヤ

Python コード

Algorithm ID: `native:randompointsinextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

入力レイヤの領域にランダム点群

指定された個数のランダムなポイントを指定されたレイヤの内部にすべて収まるように生成したポイントレイヤを新たに作成します。

ポイントが互いに近くなりすぎることを避けるため、最小距離を指定できます。

デフォルトメニュー ： ベクタ　 調査ツール

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	領域を定義する入力ポリゴンレイヤ
点の数	POINTS_NUMBER	[数値] デフォルト： 1	作成したいポイントの数
点間距離の最小値	MIN_DISTANCE	[数値] デフォルト： 0.0	ポイント間の最小距離
ランダム点群出力	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ランダム点群の出力。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ランダム点群出力	OUTPUT	[ベクタ：ポイント]	ランダムなポイントの出力レイヤ

Python コード

Algorithm ID: qgis:randompointsinlayerbounds

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ポリゴン内部にランダム点群

ポリゴンレイヤの内部にポイントが乗っているポイントレイヤを作成します。

入力レイヤ内の地物（ポリゴン / マルチポリゴン）ジオメトリそれぞれについて、結果レイヤには指定した個数のポイントが作成されます。

出力ポイントレイヤ内でポイントが近くなりすぎないように、地物ごとの最小距離とグローバルな最小距離を指定することができます。最小距離を指定した場合、地物それぞれで指定した個数のポイントを作成できない可能性があります。作成したポイントの合計数や作成できなかったポイント数は、アルゴリズムの出力から確認できます。

以下の図に、地物あたりの最小距離と全体の最小距離の違いや、最小距離をゼロ / 非ゼロとした場合の効果を示します（同一の乱数シードで生成しているので、最初に生成される点は同じです）。

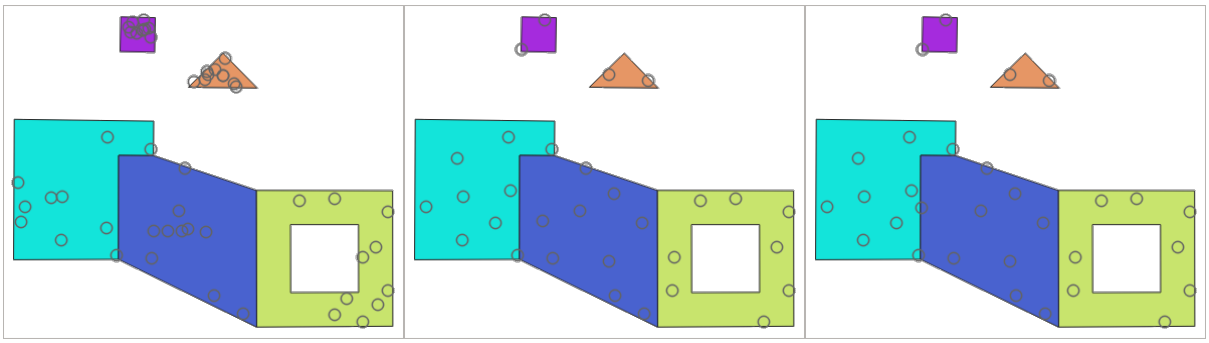


図 27.44: 1 ポリゴンあたり 10 個のポイントを生成。 **左** ：最小距離は両方とも 0、 **中央** ：最小距離は両方とも 1、 **右** ：地物あたり最小距離は 1、全体の最小距離は 0

ポイントの生成時ごとの最大試行回数を指定できます。これは最小距離に非ゼロの値を指定した場合にのみ意味があります。

乱数発生器のシードを指定して、アルゴリズムの実行時に毎回、同一の乱数列を利用することもできます。

生成するポイントに、その場所にあるポリゴン地物の属性を入れることができます（ **ポリゴンの属性を含める** ）。

全てのポリゴン地物についておおよそ等しいポイント密度で発生させたい場合には、ポリゴン地物ジオメトリの面積を使用して、ポイント数をデータ定義とするのがよいでしょう。

参考:

ポリゴン内部にランダム点群

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力ポリゴンレイヤ	INPUT	[ベクタ：ライン]	入力ポリゴンベクタレイヤ
地物あたりの点の数	POINTS_NUMBER	[数値] デフォルト： 1	作成したいポイントの数
点間距離の最小値 オプション	MIN_DISTANCE	[数値] デフォルト： 0.0	1 つのポリゴン地物内のポイント間の最小距離
ポリゴン内部にランダム点群	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ランダム点群の出力。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
点間最小距離（全体） オプション	MIN_DISTANCE_GLO	[数値] デフォルト： 0.0	全体についての点間の最小距離。このパラメータが効果を発揮するためには、**（地物あたりの）点間距離の最小値** よりも小さくする必要があります。
最大試行回数 オプション	MAX_TRIES_PER_PO	[数値] デフォルト： 10	ポイントの生成時ごとの最大試行回数。これは、点間の最小距離が設定されている（かつ 0 よりも大きい）場合にのみ意味があります。
乱数のシード オプション	SEED	[数値] デフォルト：未設定	乱数発生器に使用するシードの値
ポリゴンの属性を含める	INCLUDE_POLYGON_	[ブール値] デフォルト： True	チェックされている場合、ポイント地物はその場所にあるポリゴンの属性を取得します。

出力

ラベル	名前	データ型	説明
ポリゴン内部にランダム点群	OUTPUT	[ベクタ：ポイント]	ランダムなポイントの出力レイヤ
点が全く作成できなかった地物の数	FEATURES_WITH_EM	[数値]	
作成された点の数	OUTPUT_POINTS	[数値]	
作成できなかった点の数	POINTS_MISSED	[数値]	最小距離の制限によって生成することができなかったポイントの数
作成できなかった点がある不完全な地物の数	POLYGONS_WITH_MI	[数値]	ポイントを作成できなかったか、ポリゴンのジオメトリが無いために、内部にポイントが含まれないポリゴン地物の数

Python コード

Algorithm ID: `native:randompointsinpolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ポリゴン内部にランダム点群

入力ポリゴンレイヤの各ポリゴンの内部にそれぞれ指定された個数のランダムなポイントを発生させたポイントレイヤを新たに作成します。

二つのサンプリング基準を利用可能です：

- 点の数：各地物についてのポイントの個数
- 点の密度：各地物についてのポイントの密度

ポイントが互いに近くなりすぎることを避けるため、最小距離を指定できます。

デフォルトメニュー ： **ベクタ　 調査ツール**

参考:

ポリゴン内部にランダム点群

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
点をランダムに抽出する基準	STRATEGY	[列挙型] デフォルト： 0	使用するサンプリング基準。次のいずれかです： • 0 --- 点の数：各地物についてのポイントの個数 • 1 --- 点の密度：各地物についてのポイントの密度
点の数もしくは密度	VALUE	[数値] デフォルト： 1.0	選択した **点をランダムに抽出する基準** に応じた、ポイントの個数または密度の値
点間距離の最小値	MIN_DISTANCE	[数値] デフォルト： 0.0	ポイント間の最小距離
ランダム点群出力	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ランダム点群の出力。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ランダム点群出力	OUTPUT	[ベクタ：ポイント]	ランダムなポイントの出力レイヤ

Python コード

Algorithm ID: `qgis:randompointsinsidepolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される

ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線に沿ったランダム点群

ラインレイヤの線上にポイントが乗っているポイントレイヤを作成します。

入力レイヤ内の地物（ライン / マルチライン）ジオメトリそれぞれについて、結果レイヤには指定した個数のポイントが作成されます。

出力ポイントレイヤ内でポイントが近くなりすぎないように、地物ごとの最小距離とグローバルな最小距離を指定することができます。最小距離を指定した場合、地物それぞれで指定した個数のポイントを作成できない可能性があります。作成したポイントの合計数や作成できなかったポイント数は、アルゴリズムの出力から確認できます。

以下の図に、地物あたりの最小距離と全体の最小距離の違いや、最小距離をゼロ / 非ゼロとした場合の効果を示します（同一の乱数シードで生成しているので、最初に生成される点は同じです）。

図 27.45: 1 ラインあたり 5 個のポイントを生成。 **左** ：最小距離は両方とも 0、 **中央** ：最小距離は両方とも非 0、 **右** ：地物あたり最小距離は非 0、全体の最小距離は 0

ポイントの生成時ごとの最大試行回数を指定できます。これは最小距離に非ゼロの値を指定した場合にのみ意味があります。

乱数発生器のシードを指定して、アルゴリズムの実行時に毎回、同一の乱数列を利用することもできます。

生成するポイントに、その場所にあるライン地物の属性を入れることができます（ **線の属性を含める** ）。

全てのライン地物についておおよそ等しいポイント密度で発生させたい場合には、ライン地物ジオメトリの長さを使用して、ポイント数をデータ定義とするのがよいでしょう。

参考:

線に沿ったランダム点群

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力線レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
地物あたりの点の数	POINTS_NUMBER	[数値] デフォルト： 1	作成したいポイントの数
点間距離の最小値 オプション	MIN_DISTANCE	[数値] デフォルト： 0.0	1 つのライン地物内のポイント間の最小距離
線に沿ったランダム点群	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ランダム点群の出力。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
点間最小距離（全体） オプション	MIN_DISTANCE_GLO	[数値] デフォルト： 0.0	全体についての点間の最小距離。このパラメータが効果を発揮するためには、**（地物あたりの）点間距離の最小値** よりも小さくする必要があります。
最大試行回数 オプション	MAX_TRIES_PER_PO	[数値] デフォルト： 10	ポイントの生成時ごとの最大試行回数。これは、点間の最小距離が設定されている（かつ 0 よりも大きい）場合にのみ意味があります。
乱数のシード オプション	SEED	[数値] デフォルト：未設定	乱数発生器に使用するシードの値

次のページに続く

表 27.114 – 前のページからの続き

ラベル	名前	データ型	説明
線の属性を含める	INCLUDE_LINE_ATT	[ブール値] デフォルト： True	チェックされている場合、ポイント地物はその場所にあるポリゴンの属性を取得します。

出力

ラベル	名前	データ型	説明
線に沿ったランダム点群	OUTPUT	[ベクタ：ポイント]	ランダムなポイントの出力レイヤ
点が全く作成できなかった地物の数	FEATURES_WITH_EM	[数値]	
作成できなかった点がある不完全な地物の数	LINES_WITH_MISSE	[数値]	ポイントを作成できなかったか、ポリゴンのジオメトリが無いために、内部にポイントが含まれないポリゴン地物の数
作成された点の数	POINTS_GENERATED	[数値]	
作成できなかった点の数	POINTS_MISSED	[数値]	最小距離の制限によって生成することができなかったポイントの数

Python コード

Algorithm ID: `native:randompointsonlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのベクタ化（pixels to points）

ラスタレイヤ内の各ピクセルに対応したポイントのベクタレイヤを作成します。

ラスタレイヤの各ピクセルの中心に対してポイント地物を作成することで、ラスタレイヤをベクタレイヤへ変換します。nodata 値のピクセルは出力では省略されます。

パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
バンド番号	RASTER_BAND	[ラスタのバンド]	データを抜き出したいラスタのバンド
属性名	FIELD_NAME	[文字列] デフォルト：'VALUE'	ラスタのバンドの値を格納するフィールドの名前
点ベクタ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	結果のピクセル中心のポイントレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
点ベクタ	OUTPUT	[ベクタ：ポイント]	結果のピクセル中心のポイントレイヤ

Python コード

Algorithm ID: native:pixelstopoints

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタのベクタ化（pixels to polygons）

ラスタレイヤ内の各ピクセルに対応したポリゴンのベクタレイヤを作成します。

ラスタレイヤの各ピクセルの範囲に対してポリゴン地物を作成することで、ラスタレイヤをベクタレイヤへ変換します。nodata 値のピクセルは出力では省略されます。

パラメータ

ラベル	名前	データ型	説明
ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
バンド番号	RASTER_BAND	[ラスタのバンド]	データを抜き出したいラスタのバンド
属性名	FIELD_NAME	[文字列] デフォルト：'VALUE'	ラスタのバンドの値を格納するフィールドの名前
ポリゴンベクタ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	結果のピクセル範囲のポリゴンレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ポリゴンベクタ	OUTPUT	[ベクタ：ポリゴン]	結果のピクセル範囲のポリゴンレイヤ

Python コード

Algorithm ID: `native:pixelstopolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

規則的な点群

指定した範囲内に規則的な格子状にポイントを生成したポイントレイヤを新たに作成します。

このグリッドは、ポイントの間隔（X, Y ともに同じ幅）または生成する点の数を指定することができます。点の数を指定する場合は、ポイントの間隔は指定した範囲の大きさから決定されます。また、このアルゴリズムは完全な長方形のグリッドを作成するため、ユーザーによって指定された点の数はグリッドを構成するポイントの最小個数として扱われます。

ポイントの間隔をランダムにすることで、規則的ではないポイントパターンを生成することもできます。

デフォルトメニュー ： ベクタ　 調査ツール

パラメータ

ラベル	名前	データ型	説明
作成範囲 (xmin, xmax, ymin, ymax)	`EXTENT`	[範囲]	ランダム点群を生成するマップ範囲 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算... ：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
点の間隔/数	`SPACING`	[数値] デフォルト： 100	ポイント間の間隔、またはポイントの個数。 点の間隔を使う オプションがチェックされているかどうかによる。
左上角からの切込み量	`INSET`	[数値] デフォルト： 0.0	指定領域の左上角に対するポイント位置のオフセット量。値は X 軸、Y 軸両方に使用されます。
点の間隔をランダムにずらす	`RANDOMIZE`	[ブール値] デフォルト： False	チェックした場合、ランダムな間隔でポイントを生成します

次のページに続く

表 27.116 – 前のページからの続き

ラベル	名前	データ型	説明
点の間隔を使う	IS_SPACING	[ブール値] デフォルト： True	チェックしない場合、点の間隔を考慮しません（点の数が入力されたものとして実行します）
出力の座標参照系（CRS）	CRS	[crs] デフォルト： **プロジェクト** *CRS*	ランダム点群レイヤの CRS
規則的な点群	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	規則的な点群レイヤの出力の指定。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
規則的な点群	OUTPUT	[ベクタ：ポイント]	規則的な点群レイヤの出力

Python コード

Algorithm ID: `qgis:regularpoints`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.17 ベクター一般

投影法の割り当て

ベクタレイヤに新しい投影法を割り当てます。

これが作成するのは、入力されたものとまったく同じ地物とジオメトリを持つ新しいレイヤですが、新しい CRS に割り当てられています。ジオメトリは再投影 **されません** 。異なる CRS に割り当てられるだけです。

このアルゴリズムは、誤った投影法が割り当てられたレイヤを修復するために使用します。

属性値は、このアルゴリズムでは変更されません。

参考:

シェープファイルの投影法の定義 、 **投影法を調べる** 、 **レイヤの再投影**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	間違った CRS、または CRS の無いベクタレイヤ
割り当てられた CRS	CRS	[crs] デフォルト：EPSG:4326 - WGS84	ベクタレイヤに割り当てたい新しい CRS を選択します
割り当てられた CRS オプション	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
割り当てられた CRS	OUTPUT	[入力レイヤと同じ]	指定した投影法のベクタレイヤ

Python コード

Algorithm ID: `native:assignprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

Nominatim ジオコーディング

入力レイヤの文字列フィールドに対して、Nominatim サービスを使ったバッチジオコーディングを実行します。出力レイヤは、ジオコーディングされた場所を反映したポイントジオメトリと、ジオコーディングされた場所に関連するいくつかの属性を持ちます。

ライン地物の **地物の *In-place* 編集** が可能です

注釈: このアルゴリズムは、OpenStreetMap 財団が提供する Nominatim ジオコーディングサービスの 使用ポリシー に準拠しています。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物をジオコーディングするベクタレイヤ
住所フィールド	FIELD	[テーブルのフィールド：文字列]	ジオコーディングする住所を格納しているフィールド
出力	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ジオコーディングされた住所のみを含む出力レイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力	OUTPUT	[ベクタ：ポイント]	ジオコーディングされた住所に対応する点地物を持つベクタレイヤ

Python コード

Algorithm ID: native:batchnominatimgeocoder

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤをブックマークに変換

レイヤ内の地物の範囲に対応する空間ブックマークを作成します。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	入力ベクタレイヤ
ブックマーク先	DESTINATION	[列挙型] デフォルト： 0	ブックマークの保存先を選択します。次のいずれかです： • 0 --- プロジェクト・ブックマーク • 1 --- ユーザー・ブックマーク
ブックマーク名になる属性	NAME_EXPRESSION	[式]	作成されるブックマークの名前を表すフィールドまたは式
グループを示すフィールド	GROUP_EXPRESSION	[式]	作成されるブックマークのグループを示すフィールドまたは式

出力

ラベル	名前	データ型	説明
追加したブックマークの数	COUNT	[数値]	

Python コード

Algorithm ID: native:layertobookmarks

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ブックマークをレイヤに変換

保存済みの空間ブックマークのポリゴンを含む新しいレイヤを作成します。現在のプロジェクトに保存されたブックマーク（プロジェクト・ブックマーク）、あるいは全ユーザー用の保存されたブックマーク（ユーザー・ブックマーク）、またはその両方を出力するように選択できます。

パラメータ

ラベル	名前	データ型	説明
出力する対象	SOURCE	[列挙型] [リスト] デフォルト：[0,1]	ブックマークのソース（複数可）を選択します。次のうちの 1 つ以上を選択します： • 0 --- プロジェクト・ブックマーク • 1 --- ユーザー・ブックマーク
出力 CRS	CRS	[crs] デフォルト：EPSG:4326 - WGS84	出力レイヤの CRS
出力	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力	OUTPUT	[ベクタ：ポリゴン]	（ブックマークの）出力ベクタレイヤ

Python コード

Algorithm ID: native:bookmarkstolayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性インデックスを作成

属性テーブルのフィールドに対して、クエリを高速化するための属性インデックスを作成します。属性インデックスの作成をサポートしているかは、レイヤのデータプロバイダとフィールドの型の両方によります。

出力はありません。インデックスはレイヤ自体に保存されます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	属性インデックスを作成したいベクタレイヤを選択します
対象属性（フィールド）	FIELD	[テーブルのフィールド：任意]	ベクタレイヤのフィールド

出力

ラベル	名前	データ型	説明
インデックス作成済み	OUTPUT	[入力レイヤと同じ]	指定したフィールドにインデックスが作成された入力ベクタレイヤ

Python コード

Algorithm ID: native:createattributeindex

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

空間インデックスを作成

レイヤ内の地物へのアクセスを高速化するための、地物の空間的な位置に基づく空間インデックスを作成します。空間インデックスの作成をサポートしているかは、レイヤのデータプロバイダによります。

出力レイヤは何も作成されません。

デフォルトメニュー: **ベクタ　データ管理ツール**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ

出力

ラベル	名前	データ型	説明
インデックス作成済み	OUTPUT	[入力レイヤと同じ]	空間インデックスが作成された入力ベクタレイヤ

Python コード

Algorithm ID: native:createspatialindex

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

シェープファイルの投影法の定義

既存のシェープファイル形式のデータセットに指定した CRS（投影法）を設定します。シェープファイル形式のデータセットに prj ファイルが無いが、正しい投影法は分かっている場合に便利です。

投影法の割り当て アルゴリズムとは異なり、このアルゴリズムは入力レイヤを変更するため、新しいレイヤの出力はありません。

注釈: シェープファイルのデータセットは、指定した CRS となるよう `.prj` ファイルと `.qpj` ファイルが上書きされます。ファイルが無い場合には新たに作成されます。

デフォルトメニュー: **ベクタ　データ管理ツール**

参考:

投影法の割り当て 、 **投影法を調べる** 、 **レイヤの再投影**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	投影法の情報が欠落したベクタレイヤ
CRS	CRS	[crs]	ベクタレイヤに割り当てたい CRS を選択します

出力

ラベル	名前	データ型	説明
	INPUT	[入力レイヤと同じ]	指定した投影法の入力ベクタレイヤ

Python コード

Algorithm ID: `qgis:definecurrentprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

重複ジオメトリの削除

重複したジオメトリを見つけて削除します。

属性はチェックされないため、2 つの地物のジオメトリが同一で属性値が異なる場合、どちらか一つのみしか結果レイヤに出力されません。

参考:

ジオメトリの削除 、 *NULL* **ジオメトリの削除** 、 **属性値重複を削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	重複するジオメトリを削除したいレイヤ
クリーニング済み出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
破棄された重複レコードの数	DUPLICATE_COUNT	[数値]	破棄された重複レコードの数
クリーニング済み出力	OUTPUT	[入力レイヤと同じ]	重複するジオメトリのない出力レイヤ
保持されているレコード数	RETAINED_COUNT	[数値]	ジオメトリの重複のないレコードの数

Python コード

Algorithm ID: `native:deleteduplicategeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性値重複を削除

指定されたフィールド（複数可）を考慮して、重複する行を削除します。最初にマッチした行は残され、その後にマッチした重複行は破棄されます。

オプションとして、重複し破棄されたレコードは分析用として別の出力に保存することができます。

参考:

重複ジオメトリの削除

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力レイヤ
次の項目で重複するフィールド	FIELDS	[テーブルのフィールド：任意] [リスト]	重複を定義するフィールド。これらのフィールドの値がすべて同じである地物は、重複しているものとみなされます。
フィルタリング済み（重複なし）	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	属性値に重複のない地物の出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

次のページに続く

表 27.117 – 前のページからの続き

ラベル	名前	データ型	説明
フィルタリング済み（重複含む） オプション	DUPLICATES	[入力レイヤと同じ] デフォルト：[出力をスキップ]	出力レイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
フィルタリング済み（重複含む） オプション	DUPLICATES	[入力レイヤと同じ] デフォルト：[出力をスキップ]	削除された地物が含まれるベクタレイヤ。指定しない（[出力をスキップ] のままにする）ならば作成されません。
破棄された重複レコードの数	DUPLICATE_COUNT	[数値]	破棄された重複レコードの数
フィルタリング済み（重複なし）	OUTPUT	[入力レイヤと同じ]	属性値に重複のない地物のベクタレイヤ
保持されているレコード数	RETAINED_COUNT	[数値]	ジオメトリの重複のないレコードの数

Python コード

Algorithm ID: native:removeduplicatesbyattribute

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタレイヤを比較

2 つのベクタレイヤを比較し、両者の間で変更がない地物、追加された地物、削除された地物を判別します。同じデータセットの異なる 2 つのバージョンを比較するためのアルゴリズムです。

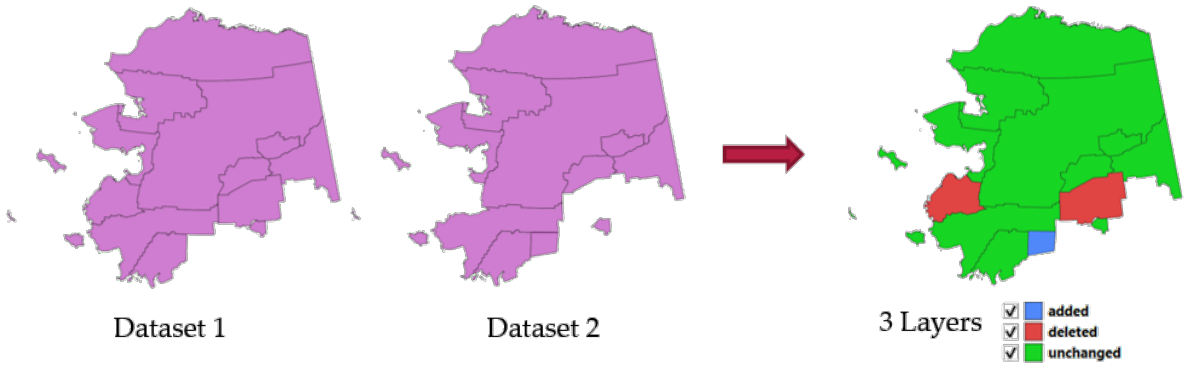


図 27.46: データセット変更の検出の例

パラメータ

ラベル	名前	データ型	説明
オリジナルレイヤ	ORIGINAL	[ベクタ：任意]	オリジナルバージョンとするベクタレイヤ
比較レイヤ	REVISED	[ベクタ：任意]	修正または変更されたベクタレイヤ
考慮する属性（ない場合はジオメトリのみ比較） オプション	COMPARE_ATTRIBUT	[テーブルのフィールド：任意] [リスト]	マッチングに考慮する属性。デフォルトでは、すべての属性値が比較されます。
ジオメトリ比較法 オプション	MATCH_TYPE	[列挙型] デフォルト： 1	比較の基準を指定します。選択肢は次のとおりです： • 0 --- 厳密マッチ：ジオメトリの線の向きや頂点の数まで含めて一致する場合にマッチ扱い • 1 --- トポロジカルマッチ：ジオメトリが等しいとみなされるものはマッチ扱い（寛容なマッチング）

次のページに続く

表 27.119 – 前のページからの続き

ラベル	名前	データ型	説明
変更がない地物 オプション	UNCHANGED	[ベクタ：元のレイヤと同じ]	変更がない地物を含む出力ベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
追加された地物 オプション	ADDED	[ベクタ：元のレイヤと同じ]	追加された地物を含む出力ベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
削除された地物 オプション	DELETED	[ベクタ：元のレイヤと同じ]	削除された地物を含む出力ベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
変更がない地物	UNCHANGED	[ベクタ：元のレイヤと同じ]	変更がない地物のベクタレイヤ
追加された地物	ADDED	[ベクタ：元のレイヤと同じ]	追加された地物のベクタレイヤ
削除された地物	DELETED	[ベクタ：元のレイヤと同じ]	削除された地物のベクタレイヤ

次のページに続く

表 27.120 – 前のページからの続き

ラベル	名前	データ型	説明
変更がない地物の数	UNCHANGED_COUNT	[数値]	変更がない地物の数
追加された地物の数	ADDED_COUNT	[数値]	比較レイヤに追加された地物の数
削除された地物の数	DELETED_COUNT	[数値]	オリジナルレイヤから削除された地物の数

Python コード

Algorithm ID: `native:detectvectorchanges`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリの削除

入力レイヤの属性テーブルの **ジオメトリのない** 単なるコピーを作成します。入力レイヤの属性テーブルはそのままです。

出力ファイルをローカルフォルダに保存する場合には、さまざまなファイルフォーマットの中から選択できます。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

重複ジオメトリの削除 、 *NULL* **ジオメトリの削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
出力レイヤ	OUTPUT	[テーブル]	ジオメトリなしの出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[テーブル]	ジオメトリなしの出力レイヤ。つまり入力レイヤの属性テーブルのコピー

Python コード

Algorithm ID: native:dropgeometries

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

SQL の実行

ソースレイヤに対して、 SQL 構文の単純な、あるいは複雑なクエリを実行します。

入力データソースは input1, input2... inputN として識別されます。単純なクエリは、 SELECT * FROM input1 のような形式です。

単純なクエリの他に、式や変数を SQL クエリ パラメータ自体の中に追加できます。これは、このアルゴリズムがプロセシングモデル内で実行される場合に、モデルの入力をクエリのパラメータとして使用したいときに特に便利です。クエリの例は、 SELECT * FROM [% @table %] のような形になり、この @table がモデル入力を識別する変数です。

クエリによる結果は、新しいレイヤとして追加されます。

参考:

SpatiaLite **を実行** 、 *PostgreSQL* **で** *SQL* **を実行**

パラメータ

ラベル	名前	データ型	説明
追加のデータソース（クエリ内では input1, .., inputN になる）	INPUT_DATASOURCE	[ベクタ：任意] [リスト]	クエリのレイヤのリスト。SQL エディタでは、これらのレイヤを **実際の** レイヤ名でも参照できますが、選択したレイヤ数に応じて **input1** 、 **input2** 、 **inputN** という名前でも参照できます。
SQL クエリ	INPUT_QUERY	[文字列]	SELECT * FROM input1 のように SQL クエリの文字列を入力します
ユニーク ID フィールド オプション	INPUT_UID_FIELD	[文字列]	ユニークな ID を持ったカラムを指定します
ジオメトリフィールド オプション	INPUT_GEOMETRY_F	[文字列]	ジオメトリフィールドを指定します
ジオメトリタイプ オプション	INPUT_GEOMETRY_T	[列挙型] デフォルト： 0	結果のジオメトリを選択します。デフォルトでは、アルゴリズムがジオメトリを自動検出します。次のいずれかです： • 0 --- 自動検出 • 1 --- ジオメトリなし • 2 --- Point • 3 --- LineString • 4 --- Polygon • 5 --- MultiPoint • 6 --- MultiLineString • 7 --- MultiPolygon

次のページに続く

表 27.121 – 前のページからの続き

ラベル	名前	データ型	説明
CRS オプション	INPUT_GEOMETRY_C	[crs]	出力レイヤに割り当てる CRS
出力レイヤ	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	クエリによって作成される出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：任意]	クエリによって作成されるベクタレイヤ

Python コード

Algorithm ID: `qgis:executesql`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤを DXF にエクスポート

レイヤを DXF ファイルにエクスポートします。各レイヤについてフィールドを選択し、その値によって DXF 出力の生成された出力先レイヤに地物を分けることができます。

参考:

新しい *DXF* ファイルを作成する

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	LAYERS	[ベクタ：任意] [リスト]	エクスポートしたい入力ベクタレイヤ
シンボロジーモード	SYMBOLOGY_MODE	[列挙型] デフォルト： 0	出力レイヤに適用するシンボロジの種類。次から選ぶことができます: • 0 -- シンボロジなし • 1 -- 地物シンボロジ • 2 -- シンボルレイヤシンボロジ
シンボルのスケール	SYMBOLOGY_SCALE	[縮尺] デフォルト: 1:1 000 000	デフォルトのエクスポートするデータの縮尺
文字コード	ENCODING	[列挙型]	レイヤに適用する文字コード
CRS	CRS	[crs]	出力レイヤの CRS を選びます
レイヤ名を名前に使用	USE_LAYER_TITLE	[ブール値] デフォルト：False	レイヤ名の代わりに（QGIS で付けられた）レイヤタイトルを使って出力レイヤに名前を付けます。
二次元出力を強制	FORCE_2D	[ブール値] デフォルト：False	
ラベルを MTEXT 要素としてエクスポートする	MTEXT	[ブール値] デフォルト：False	ラベルを MTEXT または TEXT 要素としてエクスポートします
DXF	OUTPUT	[ファイル] デフォルト：[一時レイヤを作成]	DXF 出力ファイルを指定します。次のいずれかです： • 一時レイヤを作成 • ファイルに保存...

出力

ラベル	名前	データ型	説明
DXF	OUTPUT	[ファイル]	.DXF ファイルは入力レイヤを持ちます

Python コード

Algorithm ID: `native:dxfexport`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

選択した地物で別レイヤ

選択された地物を新しいレイヤとして保存します。

注釈: 選択されたレイヤに選択された地物がない場合には、新しく作成されるレイヤは空のレイヤです。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	選択地物を保存したいレイヤ
選択されている地物	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	選択地物によるベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
選択されている地物	OUTPUT	[入力レイヤと同じ]	選択地物のみのベクタレイヤ。地物選択がない場合には地物のないレイヤ。

Python コード

Algorithm ID: `native:saveselectedfeatures`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

シェープファイルの文字コードを抽出

シェープファイルに埋め込まれた属性値の文字コード情報を抽出します。オプションファイルの `.cpg` ファイルで指定される文字コードと、 `.dbf` ファイルの LDID ヘッダブロックにある文字コード詳細の両方がチェックされます。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	`INPUT`	[ベクタ：任意]	文字コード情報を抽出したい ESRI シェープファイル（ `.SHP` ）レイヤ

出力

ラベル	名前	データ型	説明
文字コード	`ENCODING`	[文字列]	入力ファイル内で指定されている文字コードの情報
CPG ファイルの文字コード	`CPG_ENCODING`	[文字列]	オプションファイルの `.CPG` ファイル内で指定されている文字コード情報
LDID 部分の文字コード	`LDID_ENCODING`	[文字列]	`.dbf` ファイルの LDID ヘッダブロックで指定されている文字コード情報

Python コード

Algorithm ID: `native:shpencodinginfo`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

投影法を調べる

例えば投影法が不明なレイヤのために、座標参照系の候補のリストを作成します。

レイヤがカバーする領域は「レイヤのターゲット領域」パラメータで指定する必要があります。このターゲット領域の座標参照系は QGIS に渡す必要があります。

このアルゴリズムは、レイヤの範囲を既知のすべての座標参照系でテストし、レイヤがこの投影法であると仮定するとその境界がターゲット領域の近くになる可能性があるものをすべてリスト化します。

参考:

投影法の割り当て 、 **シェープファイルの投影法の定義** 、 **レイヤの再投影**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	投影法が不明なレイヤ

次のページに続く

表 27.123 – 前のページからの続き

ラベル	名前	データ型	説明
レイヤのターゲット領域 (xmin, xmax, ymin, ymax)	TARGET_AREA	[範囲]	レイヤがカバーする領域 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
出力レイヤ	OUTPUT	[テーブル] デフォルト：[一時レイヤを作成]	CRS の提案（EPSG コード）をリストしたテーブル（ジオメトリなしのレイヤ）の出力を指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[テーブル]	基準にマッチするすべての CRS（EPSG コード）をリストしたテーブル

Python コード

Algorithm ID: `qgis:findprojection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

リレーションのフラット化

ベクタレイヤの **リレーション** のフラット化、すなわち、関連する子地物ごとに 1 つの親地物を含んだ単一のレイヤを作成します。このマスタ地物には、関連する地物のすべての属性が含まれます。これにより、リレーションを単純なテーブルとして持たせられ、例えば CSV にエクスポートすることができます。

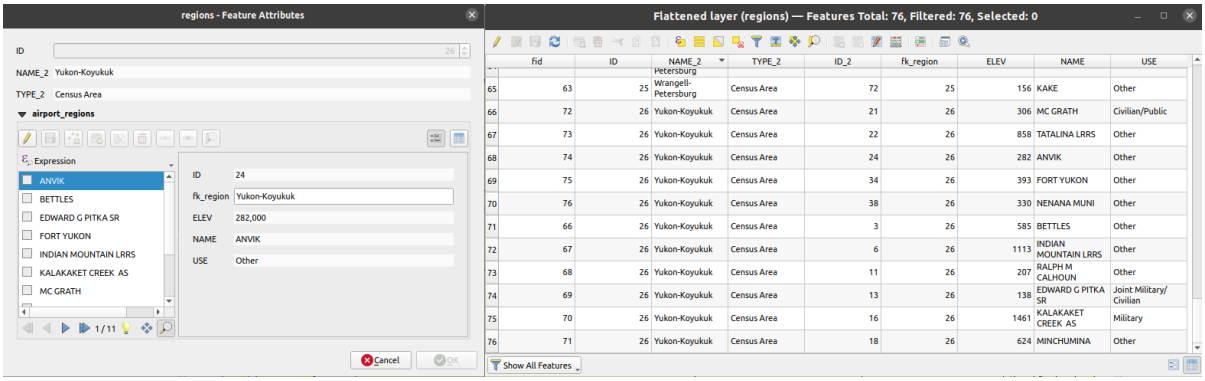


図 27.47: region のフォームと、リレーションのある子地物（左）— リレーションのある子地物ごとに複製された region の地物と、結合された属性（右）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	リレーションの非正規化を行いたいレイヤ

次のページに続く

表 27.124 – 前のページからの続き

ラベル	名前	データ型	説明
フラット化出力オプション	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力レイヤ（フラット化されたレイヤ）を指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
フラット化出力	OUTPUT	[入力レイヤと同じ]	マスタ地物と、リレーションのある地物の全ての属性を含むレイヤ

Python コード

Algorithm ID: native:flattenrelationships

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性テーブルで結合（table join）

入力ベクタレイヤを受け取り、その属性テーブルに別の属性を追加して、入力レイヤの拡張版となる新しいベクタレイヤを作成します。

追加の属性とその値は、第 2 のベクタレイヤから取得されます。それぞれのレイヤで属性を選択し、結合の条件が定義されます。

参考:

属性の最近傍結合 、 **属性の空間結合**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ。出力レイヤはこのレイヤの地物と、第２のレイヤでマッチする地物の属性からなります。
入力レイヤの結合対象フィールド	FIELD	[テーブルのフィールド：任意]	ソースレイヤで結合に使用するフィールド
第２の入力レイヤ	INPUT_2	[ベクタ：任意]	属性テーブルを結合させたいレイヤ
第２の入力レイヤの結合対象フィールド	FIELD_2	[テーブルのフィールド：任意]	結合に使用する第２のレイヤ（結合レイヤ）のフィールド。このフィールドの型は、入力テーブルのフィールド型と等しい（または互換性がある）ものでなければなりません。
第２の入力レイヤからコピーする属性(全属性をコピーする場合は空のまま) オプション	FIELDS_TO_COPY	[テーブルのフィールド：任意] [リスト]	追加したいフィールドを指定して選択します。デフォルトでは、すべてのフィールドが追加されます。
結合のタイプ	METHOD	[列挙型] デフォルト： 1	レイヤの結合のタイプ。次のいずれかです： • 0 --- マッチした地物ごとに地物を作成（１対多結合） • 1 --- 最初に合致した地物の属性のみを取得（１対１結合）
結合対象がなかった地物を破棄	DISCARD_NONMATCH	[ブール値] デフォルト： True	結合できなかった地物を残したくない場合には、チェックを入れてください
コピーしたフィールドの接頭辞 オプション	PREFIX	[文字列]	結合したフィールドを区別しやすくし、フィールド名の衝突を避けるため、結合フィールドに接頭辞を追加します
出力レイヤ オプション	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	結合したベクタレイヤの出力を指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

次のページに続く

表 27.125 – 前のページからの続き

ラベル	名前	データ型	説明
入力レイヤのうち、結合対象がなかった地物 オプション	NON_MATCHING	[入力レイヤと同じ] デフォルト：[出力をスキップ]	最初のレイヤの地物のうち、結合できなかった地物のベクタレイヤの出力を指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
入力テーブルから結合された地物の数	JOINED_COUNT	[数値]	
入力レイヤのうち、結合対象がなかった地物 オプション	NON_MATCHING	[入力レイヤと同じ]	結合できなかった地物のベクタレイヤ
出力レイヤ オプション	OUTPUT	[入力レイヤと同じ]	結合レイヤの属性が追加された出力ベクタレイヤ
入力テーブルから結合できなかった地物の数 オプション	UNJOINABLE_COUNT	[数値]	

Python コード

Algorithm ID: native:joinattributestable

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性の空間結合

入力ベクタレイヤを受け取り、その属性テーブルに別の属性を追加して、入力レイヤの拡張版となる新しいベクタレイヤを作成します。

追加される属性とその値は、第 2 のベクタレイヤから取得されます。空間的な基準を適用して選択された第 2 のレイヤの値が、第 1 のレイヤの各地物に追加されます。

デフォルトメニュー: **ベクタ　データ管理ツール**

参考:

属性の最近傍結合 、 **属性テーブルで結合（*table join*）** 、 **空間結合（集計つき）**

空間的関係を調べる

幾何学的述語は、ある地物が他の地物と空間の一部を共有しているかどうか、またどのように共有しているかを比較することによって、空間的関係を持つかどうかを決定するために使用されるブール関数です。

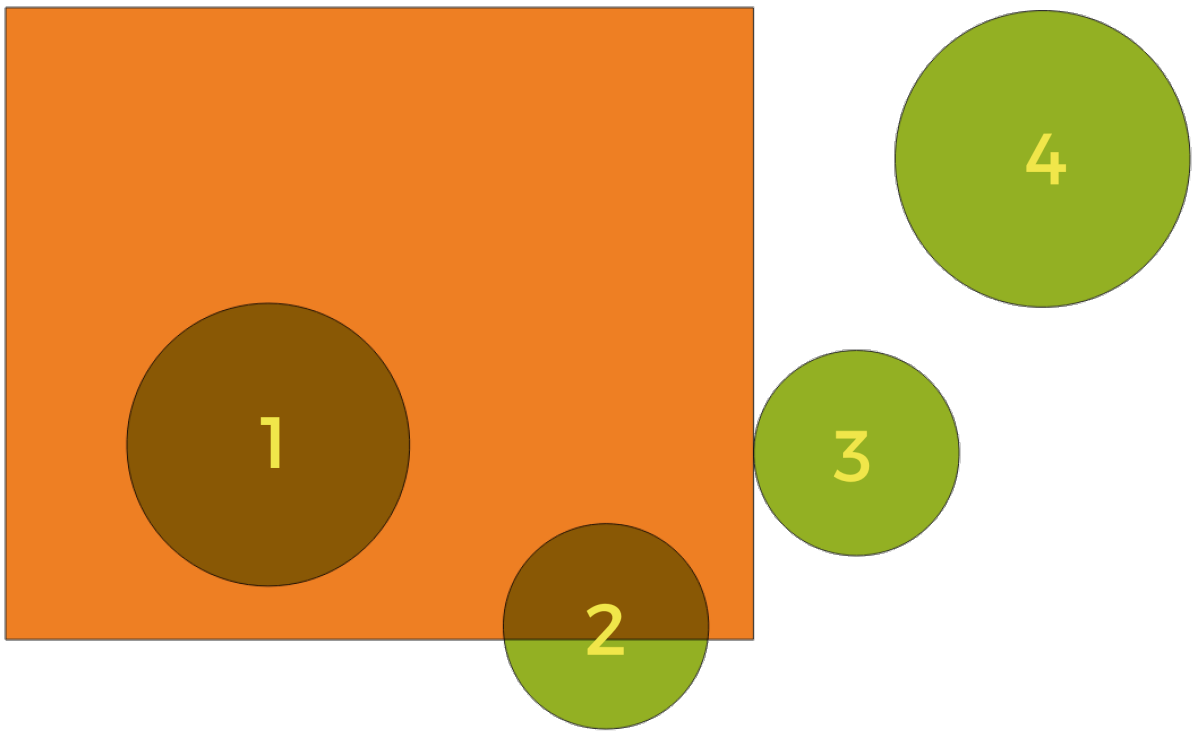


図 27.48: レイヤ間の空間的関係を求める

上の図を使って、オレンジ色の長方形の地物と空間的に比較することで、緑色の円を求めます。利用可能な幾何学的述語は以下の通りです：

交差する *(intersect)*
: あるジオメトリが他のジオメトリと交差しているかどうかをテストします。ジオメトリが空間的に交差している（空間の一部を共有している - 重なり合うか接触している）場合は 1（true）を返し、交差していない場合は 0 を返す。上の図では、円 1、2、3 を返す。

含む *(contain)*

b の点が a の外側になく、かつ b の内側の少なくとも 1 点が a の内側にある場合にのみ、1 (true) を返す。この図では、どの円も返さないが、逆を探せば、長方形は円 1 を完全に含むので、返す。これは **含まれる** *(within)* の逆である。

離れている *(disjoint)*

ジオメトリが空間のどの部分も共有していない（重なっていない、接触していない）場合、1 (true) を返す。円 4 のみが返される。

等しい *(equal)*

ジオメトリが完全に同じ場合のみ、1 (true) を返す。どの円も返されない。

接触する *(touch)*

あるジオメトリが別のジオメトリに接しているかどうかを調べます。ジオメトリに少なくとも 1 つの共通点があるが、内部が交差していない場合は 1 (true) を返します。円 3 のみが返されます。

重なる *(overtap)*

あるジオメトリが別のジオメトリに重なっているかどうかを調べます。ジオメトリが空間を共有し、同じ大きさであるが、互いに完全に含まれない場合は 1 (true) を返します。円 2 のみが返されます。

含まれる *(within)*

あるジオメトリが他のジオメトリの中にあるかどうかを調べます。ジオメトリ a が完全にジオメトリ b の内側にある場合は 1 (true) を返します。

交差する *(cross)*

与えられたジオメトリが、すべてではなく、いくつかの内部点を共通に持ち、交差部が与えられたジオメトリの最大よりも小さい次元の場合、1 (true) を返します。例えば、ポリゴンを横切るラインは、ラインとして交差します (true)。交差する 2 本のラインは、ポイントとして交差します (true)。2 つのポリゴンはポリゴンとして交差します (false)。図では、どの円は返されません。

パラメータ

ラベル	名前	データ型	説明
地物を結合するレイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ。出力レイヤはこのレイヤの地物と、第 2 のレイヤでマッチする地物の属性からなります。

次のページに続く

表 27.126 – 前のページからの続き

ラベル	名前	データ型	説明
空間的関係	PREDICATE	[列挙型] [リスト] デフォルト： [0]	ソース地物がターゲット地物との間に持っている場合に結合される、空間関係の種類。次の 1 つ以上： • 0 -- 交差する (intersect) • 1 -- 含む (contain) • 2 -- 等しい (equal) • 3 -- 接触する (touch) • 4 -- 重なる (overlap) • 5 -- 含まれる (within) • 6 -- 交差する (cross) 複数の条件が選択された場合、そのうちの少なくとも 1 つ（OR 演算）を満たさなければ地物は抽出されません。
比較対象	JOIN	[ベクタ：任意]	結合レイヤ。このベクタレイヤの地物の空間的関係が満たされるとき、その属性がソースレイヤの属性テーブルに **追加** されます。
結合するフィールド（すべてのフィールドを結合する場合は空のまま） オプション	JOIN_FIELDS	[テーブルのフィールド：任意] [リスト]	結合レイヤから追加したい特定のフィールドを選択します。デフォルトでは、すべてのフィールドが追加されます。
結合のタイプ	METHOD	[列挙型]	レイヤの結合のタイプ。次のいずれかです： • 0 --- マッチした地物ごとに地物を作成（1 対多結合） • 1 --- 最初に合致した地物の属性のみを取得（1 対 1 結合） • 2 --- もっとも重なる地物の属性のみ（1 対 1）
結合対象がなかった地物を破棄	DISCARD_NONMATCH	[ブール値] デフォルト： False	結合できなかった入力レイヤの地物を出力結果では削除します
コピーしたフィールドの接頭辞 オプション	PREFIX	[文字列]	結合したフィールドを区別しやすくし、フィールド名の衝突を避けるため、結合フィールドに接頭辞を追加します

次のページに続く

表 27.126 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ オプション	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	結合したベクタレイヤの出力を指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
入力レイヤのうち、結合対象がなかった地物 オプション	NON_MATCHING	[入力レイヤと同じ] デフォルト：[出力をスキップ]	最初のレイヤの地物のうち、結合できなかった地物のベクタレイヤの出力を指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
入力テーブルから結合された地物の数	JOINED_COUNT	[数値]	
入力レイヤのうち、結合対象がなかった地物 オプション	NON_MATCHING	[入力レイヤと同じ]	結合できなかった地物のベクタレイヤ
出力レイヤ	OUTPUT	[入力レイヤと同じ]	結合レイヤの属性が追加された出力ベクタレイヤ

Python コード

Algorithm ID: `native:joinattributesbylocation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

空間結合（集計つき）

入力ベクタレイヤを受け取り、その属性テーブルに別の属性を追加して、入力レイヤの拡張版となる新しいベクタレイヤを作成します。

追加される属性とその値は、第 2 のベクタレイヤから取得されます。空間的な基準を適用して選択された第 2 のレイヤの値が、第 1 のレイヤの各地物に追加されます。

このアルゴリズムは、マッチングする第 2 のレイヤの地物の値について、要約統計量（最大値、平均値など）を計算します。

参考:

属性の空間結合

空間的関係を調べる

幾何学的述語は、ある地物が他の地物と空間の一部を共有しているかどうか、またどのように共有しているかを比較することによって、空間的関係を持つかどうかを決定するために使用されるブール関数です。

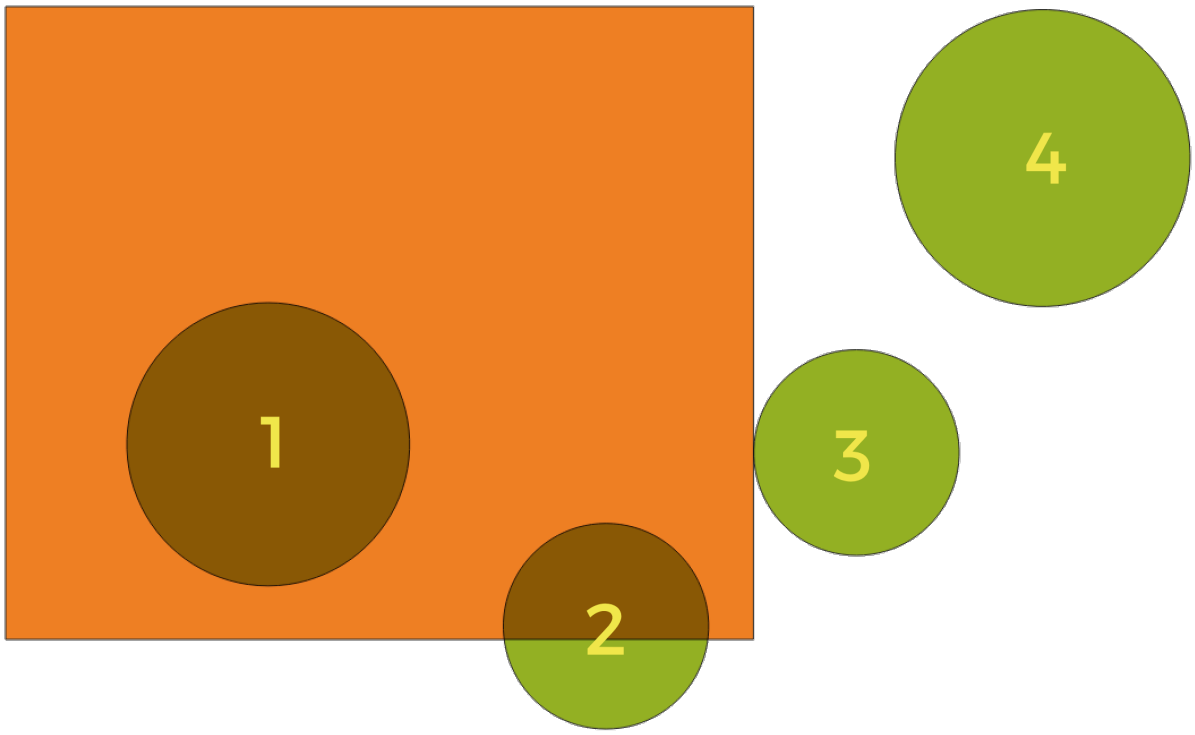


図 27.49: レイヤ間の空間的関係を求める

上の図を使って、オレンジ色の長方形の地物と空間的に比較することで、緑色の円を求めます。利用可能な幾何学的述語は以下の通りです：

交差する *(intersect)*
: あるジオメトリが他のジオメトリと交差しているかどうかをテストします。ジオメトリが空間的に交差している（空間の一部を共有している - 重なり合うか接触している）場合は 1（true）を返し、交差していない場合は 0 を返す。上の図では、円 1、2、3 を返す。

含む *(contain)*
: b の点が a の外側になく、かつ b の内側の少なくとも 1 点が a の内側にある場合にのみ、1 (true) を返す。この図では、どの円も返さないが、逆を探せば、長方形は円 1 を完全に含むので、返す。これは **含まれる** *(within)* の逆である。

離れている *(disjoint)*
: ジオメトリが空間のどの部分も共有していない（重なっていない、接触していない）場合、1 (true) を返す。円 4 のみが返される。

等しい *(equal)*
: ジオメトリが完全に同じ場合のみ、1 (true) を返す。どの円も返されない。

接触する *(touch)*
: あるジオメトリが別のジオメトリに接しているかどうかを調べます。ジオメトリに少なくとも 1 つの共通点があるが、内部が交差していない場合は 1 (true) を返します。円 3 のみが返されます。

重なる *(overtap)*
: あるジオメトリが別のジオメトリに重なっているかどうかを調べます。ジオメトリが空間を共有し、同じ大きさであるが、互いに完全に含まれない場合は 1 (true) を返します。円 2 のみが返されます。

含まれる *(within)*

あるジオメトリが他のジオメトリの中にあるかどうかを調べます。ジオメトリ a が完全にジオメトリ b の内側にある場合は 1 (true) を返します。

交差する *(cross)*

与えられたジオメトリが、すべてではなく、いくつかの内部点を共通に持ち、交差部が与えられたジオメトリの最大よりも小さい次元の場合、1 (true) を返します。例えば、ポリゴンを横切るラインは、ラインとして交差します (true)。交差する 2 本のラインは、ポイントとして交差します (true)。2 つのポリゴンはポリゴンとして交差します (false)。図では、どの円は返されません。

パラメータ

ラベル	名前	データ型	説明
地物を結合するレイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ。出力レイヤはこのレイヤの地物と、第 2 のレイヤでマッチする地物の属性からなります。
空間的関係	PREDICATE	[列挙型] [リスト] デフォルト： [0]	ソース地物がターゲット地物との間に持っている場合に結合される、空間関係の種類。次の 1 つ以上： • 0 -- 交差する (intersect) • 1 -- 含む (contain) • 2 -- 等しい (equal) • 3 -- 接触する (touch) • 4 -- 重なる (overlap) • 5 -- 含まれる (within) • 6 -- 交差する (cross) 複数の条件が選択された場合、そのうちの少なくとも 1 つ（OR 演算）を満たさなければ地物は抽出されません。
比較対象	JOIN	[ベクタ：任意]	結合レイヤ。このベクタレイヤの地物の空間的関係が満たされるとき、その属性の集計がソースレイヤの属性テーブルに **追加** されます。
集計する属性（全属性の場合は空のまま） オプション	JOIN_FIELDS	[テーブルのフィールド：任意] [リスト]	結合レイヤから追加したい特定のフィールドを選択します。デフォルトでは、すべてのフィールドが追加されます。

次のページに続く

表 27.127 – 前のページからの続き

ラベル	名前	データ型	説明
計算する集計関数（すべて計算する場合は空のまま） オプション	SUMMARIES	[列挙型] [リスト] デフォルト： []	各入力地物について、マッチする地物の結合フィールドの統計が計算されます。次のひとつ以上です： • 0 --- 個数 • 1 --- ユニーク数 • 2 --- 最小値 • 3 --- 最大値 • 4 --- 範囲 • 5 --- 合計 • 6 --- 平均 • 7 --- 中央値 • 8 --- 標準偏差 • 9 --- 最稀値 • 10 --- 最頻値 • 11 --- 第１四分位（Q1） • 12 --- 第３四分位（Q3） • 13 --- 四分位範囲（iQR） • 14 --- 空ジオメトリ • 15 --- 穴がない（filled） • 16 --- 最短の長さ • 17 --- 最大の長さ • 18 --- 平均の長さ
結合対象がなかった地物を破棄	DISCARD_NONMATCH	[ブール値] デフォルト： False	結合できなかった入力レイヤの地物を出力結果では削除します
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	結合したベクタレイヤの出力を指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	結合レイヤの属性の統計量を持つ出力レイヤ

Python コード

Algorithm ID: `qgis:joinbylocationsummary`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性の最近傍結合

入力ベクタレイヤを受け取り、その属性テーブルにフィールドを追加した新しいベクタレイヤを作成します。追加される属性と値は 2 つ目のベクタレイヤから取得されます。地物は、お互いのレイヤで最も近い地物を見つけて結合します。

デフォルトでは最も近い地物のみが結合しますが、結合は近くの k 個の地物と行うこともできます。

最大距離を指定した場合には、この距離よりも近い地物のみがマッチします。

参考:

最近傍解析 、 **属性テーブルで結合（*table join*）** 、 **属性の空間結合** 、 **距離行列（*distance matrix*）**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力レイヤ
第 2 の入力レイヤ	INPUT_2	[ベクタ：任意]	結合レイヤ
第 2 の入力レイヤからコピーする属性（全属性をコピーする場合は空のまま）	FIELDS_TO_COPY	[フィールド]	コピーしたい結合レイヤのフィールド（空の場合には、すべてのフィールドがコピーされます）

次のページに続く

表 27.128 – 前のページからの続き

ラベル	名前	データ型	説明
結合対象がなかった地物を破棄	DISCARD_NONMATCH	[ブール値] デフォルト：False	結合できなかった入力レイヤのレコードを出力結果では削除します
コピーしたフィールドの接頭辞	PREFIX	[文字列]	コピーしたフィールドの接頭辞
近接地物の個数 (n)	NEIGHBORS	[数値] デフォルト：1	最近傍の個数の最大値
最大距離	MAX_DISTANCE	[数値]	検索距離の最大値
出力レイヤ オプション	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	結合した地物を含むベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
入力レイヤのうち、結合対象がなかった地物	NON_MATCHING	[入力レイヤと同じ] デフォルト：[出力をスキップ]	結合できなかった地物を含むベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	結合した出力レイヤ
入力レイヤのうち、結合対象がなかった地物	NON_MATCHING	[入力レイヤと同じ]	結合レイヤのどの地物とも結合できなかった最初のレイヤの地物を含むレイヤ
入力テーブルから結合された地物の数	JOINED_COUNT	[数値]	入力テーブルから結合された地物の数

次のページに続く

表 27.129 – 前のページからの続き

ラベル	名前	データ型	説明
入力テーブルから結合できなかった地物の数	UNJOINABLE_COUNT	[数値]	入力テーブルから結合できなかった地物の数

Python コード

Algorithm ID: native:joinbynearest

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

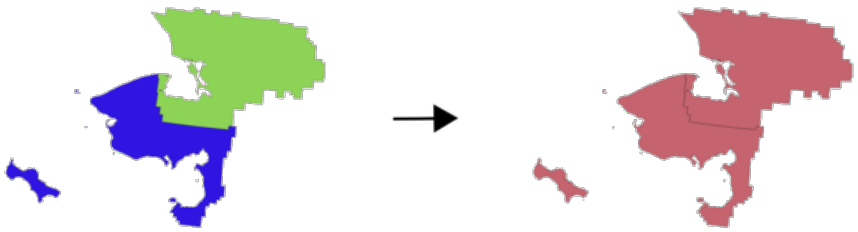
ベクタレイヤのマージ

同じジオメトリタイプ の複数のベクタレイヤを単一のベクタレイヤに統合します。

結果レイヤの属性テーブルには、すべての入力レイヤからのフィールドが含まれます。同じ名前であるが異なる型のフィールドがある場合には、出力されるフィールドは自動的に文字列型のフィールドに変換されます。元のレイヤ名とソースを保存する新しいフィールドも追加されます。

入力レイヤに Z 座標や M 値がある場合には、出力レイヤにも Z 座標や M 値が含まれます。同様に、任意の入力レイヤがマルチパートの場合には、出力レイヤもマルチパートレイヤとなります。

オプションとして、マージしたレイヤの変換先座標参照系（CRS）を設定することができます。これを設定しない場合には、CRS は最初の入力レイヤから取得されます。すべての入力レイヤはこの CRS に一致するように再投影されます。



デフォルトメニュー: **ベクタ　データ管理ツール**

参考:

属性でレイヤを分割

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	LAYERS	[ベクタ：任意] [リスト]	単一のレイヤにマージしたいレイヤ群。レイヤは同じジオメトリタイプである必要があります。
変換先の座標参照系（CRS） オプション	CRS	[crs]	出力レイヤの CRS を選択します。指定しない場合には、最初の入力レイヤの CRS が使用されます。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	入力レイヤのすべての地物と属性を持つ出力ベクタレイヤ

Python コード

Algorithm ID: native:mergevectorlayers

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

式による並べ替え

式に従ってベクタレイヤをソートします。つまり、式に従って地物のインデックスを変更します。

データプロバイダによっては順序が毎回維持されない場合があり、期待通りに機能しない可能性があるため注意して下さい。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	ソートしたい入力ベクタレイヤ
式	EXPRESSION	[式]	ソートに使用する式
昇順	ASCENDING	[ブール値] デフォルト： True	チェックを入れた場合には、ベクタレイヤは小さい値から大きい値となるように並べられます。
NULL は最初にソートされる	NULLS_FIRST	[ブール値] デフォルト：False	チェックを入れた場合には、NULL 値が最初に並びます。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（ソートされた）出力ベクタレイヤ

Python コード

Algorithm ID: native:orderbyexpression

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

シェープファイルを修復

SHX ファイルを（再度）作成することによって、壊れた ESRI シェープファイルデータセットを修復します。

パラメータ

ラベル	名前	データ型	説明
シェープファイル	INPUT	[ファイル]	SHX ファイルが無い、あるいは SHX ファイルが壊れた ESRI シェープファイルデータセットのフルパス

出力

ラベル	名前	データ型	説明
修復済みファイル	OUTPUT	[ベクタ：任意]	入力ベクタレイヤの SHX ファイルが修復されたレイヤ

Python コード

Algorithm ID: native:repairshapefile

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤの再投影

ベクタレイヤを別の CRS に再投影します。再投影されたレイヤは、入力レイヤと同じ地物と同じ属性を持ちます。

☑ ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

投影法の割り当て 、 **シェープファイルの投影法の定義** 、 **投影法を調べる**

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	再投影したい入力ベクタレイヤ
ラスタの CRS	TARGET_CRS	[crs] デフォルト：EPSG:4326 - WGS84	変換先の座標参照系
再投影したラスタファイル	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
座標演算オプション	OPERATION	[文字列]	現在のプロジェクトの変換設定を常に強制的に使用する代わりの、特定の再投影タスクに使用するための特定の操作です。特定のレイヤを再投影する際に、正確な変換パイプラインによる再投影が必要な場合に便利です。proj のバージョン 6 以上が必要です。 詳細は **測地系変換** を参照してください。

出力

ラベル	名前	データ型	説明
再投影したラスタファイル	OUTPUT	[入力レイヤと同じ]	（再投影された）出力ベクタレイヤ

Python コード

Algorithm ID: native:reprojectlayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタ地物をファイルに保存

ベクタ地物を指定したファイルデータセットに保存します。

レイヤをサポートするデータセット形式では、オプションとしてレイヤ名パラメータにカスタム文字列を指定することができます。また、オプションとして、GDAL で定義されているデータセット作成オプションやレイヤ作成オプションを指定することもできます。これに関する詳細については、その形式に関するオンラインの GDAL documentation を参照してください。

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
ベクタ地物	INPUT	[ベクタ：任意]	入力ベクタレイヤ
保存先出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	地物を保存するファイルを指定します。 次のいずれかです： • 一時レイヤを作成 • ファイルに保存...

詳細パラメータ

ラベル	名前	データ型	説明
レイヤ名 オプション	LAYER_NAME	[文字列]	出力レイヤに使用するレイヤ名
GDAL データセットオプション オプション	DATASOURCE_OPTIO	[文字列]	出力ファイル形式に関する GDAL データセット作成オプション。複数ある場合は各オプションをセミコロンで区切ります。
GDAL レイヤオプション オプション	LAYER_OPTIONS	[文字列]	出力ファイル形式に関する GDAL レイヤ作成オプション。複数ある場合は各オプションをセミコロンで区切ります。

出力

ラベル	名前	データ型	説明
保存先出力	OUTPUT	[入力レイヤと同じ]	地物が保存されたベクタレイヤ
ファイルとパス	FILE_PATH	[文字列]	出力ファイルのファイル名とパス
レイヤ名	LAYER_NAME	[文字列]	レイヤ名（あれば）

Python コード

Algorithm ID: native:savefeatures

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

レイヤの文字コードを設定

レイヤの属性値の読み込みに使用する文字コードを設定します。レイヤに変更を行うのではなく、現在のセッションにおけるレイヤの読み込みのみが影響します。

注釈: 文字コードの変更をサポートするのは、一部のベクタレイヤデータソースのみです。

パラメータ

ラベル	名前	データ型	説明
保存先出力	INPUT	[ベクタ：任意]	文字コードを設定したいベクタレイヤ
文字コード	ENCODING	[文字列]	現在の QGIS セッションにおいてレイヤに適用されるテキストエンコーディング

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	設定された文字コードの入力ベクタレイヤ

Python コード

Algorithm ID: `native:setlayerencoding`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

文字で地物を分割

フィールドの値を指定した区切り文字で分割することで、地物を複数に分割します。例えば、あるレイヤの単一のフィールドに複数のカンマ区切りの値を含む地物がある場合に、このアルゴリズムを使用してそれらの値を分割し、複数の出力地物に分割することができます。ジオメトリやその他の属性値はそのままです。オプションとして、区切り文字に正規表現を使うことができ、区切り文字を柔軟に指定できます。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
分割するフィールド	FIELD	[テーブルのフィールド：任意]	分割に使用するフィールド
区切り文字	CHAR	[文字列]	分割に使用する区切り文字
区切り文字に正規表現を使う	REGEX	[ブール値] デフォルト：False	
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	入力ベクタレイヤ

Python コード

Algorithm ID: `native:splitfeaturesbycharacter`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性でレイヤを分割

入力レイヤと属性に基づいて、出力フォルダにベクタレイヤの組を作成します。出力フォルダには、指定したフィールドで見つかったユニーク値の個数分のレイヤが作成されます。

生成されるファイルの数は、指定した属性に見つかった相異なる値の数と同じです。

これは、 **ベクタレイヤのマージ** とは反対の操作です。

デフォルトメニュー: **ベクタ　データ管理ツール**

参考:

ベクタレイヤのマージ

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
ユニーク ID 属性	FIELD	[テーブルのフィールド：任意]	分割に使用するフィールド
出力フォルダ	OUTPUT	[フォルダ] デフォルト：[一時フォルダに保存]	出力レイヤの保存ディレクトリを指定します。次のいずれかです： • 一時ディレクトリに保存 • ディレクトリに保存します

詳細パラメータ

ラベル	名前	データ型	説明
出力ファイル型 オプション	FILE_TYPE	[列挙型] デフォルト：ダイアログウィンドウで gpkg	出力ファイルの拡張子を選択します。未指定または無効の場合、出力ファイル形式はプロセシング設定の「デフォルトのベクタレイヤ拡張子」で設定されたものになります。

出力

ラベル	名前	データ型	説明
出力フォルダ	OUTPUT	[フォルダ]	出力レイヤの保存ディレクトリ
出力レイヤ	OUTPUT_LAYERS	[入力レイヤと同じ] [リスト]	分割による結果の出力ベクタレイヤ

Python コード

Algorithm ID: native:splitvectorlayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

テーブルを空にする

レイヤ内のすべての地物を削除することで、レイヤのテーブルを空にします。

> **警告:** このアルゴリズムはレイヤを直接編集し、削除された地物は元に戻すことはできません。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ

出力

ラベル	名前	データ型	説明
空のレイヤ	OUTPUT	[フォルダ]	削除された（空の）レイヤ

Python コード

Algorithm ID: `native:truncatetable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.18 ベクタジオメトリ

ジオメトリ属性を追加

ベクタレイヤの地物のジオメトリのプロパティを計算し、出力レイヤにそれらを含めて出力します。

このアルゴリズムは入力ベクタレイヤと同じ内容の新しいベクタレイヤを作成しますが、選択した CRS に基づいた幾何学的な測量値を追加の属性として持ちます。

テーブルに追加される属性は、入力レイヤのジオメトリタイプとジオメトリの次元によって異なります。

- **ポイント** レイヤの場合： X (`xcoord`), Y (`ycoord`), Z (`zcoord`) 座標、M 値 (`mvalue`)
- **ライン** レイヤの場合：長さ `length` と、ラインストリング (LineString) や複合カーブ (CompoundCurve) ジオメトリタイプの場合には地物の湾曲度 `sinuosity` と直線距離 (`straightdis`)
- **ポリゴン** レイヤの場合：周長 `perimeter` と面積 `area`

デフォルトメニュー ： **ベクタ　ジオメトリツール**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
計算に利用する座標参照系（CRS）	CALC_METHOD	[列挙型] デフォルト： 0	ジオメトリのプロパティの計算に使用するパラメータ。次のいずれかです： • 0 --- レイヤの CRS • 1 --- プロジェクトの CRS • 2 --- 回転楕円体
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力結果レイヤ（入力レイヤのコピーにジオメトリの属性値を持たせたもの）を指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	入力ベクタレイヤのコピーに追加でジオメトリのフィールドを持たせたレイヤ

Python コード

Algorithm ID: `qgis:exportaddgeometrycolumns`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

アフィン変換

レイヤのジオメトリにアフィン変換を適用します。アフィン変換には平行移動、拡大縮小、回転があります。演算は、拡大縮小、回転、平行移動の順で行われます。

（あれば）Z 座標と M 値も平行移動や拡大縮小ができます。

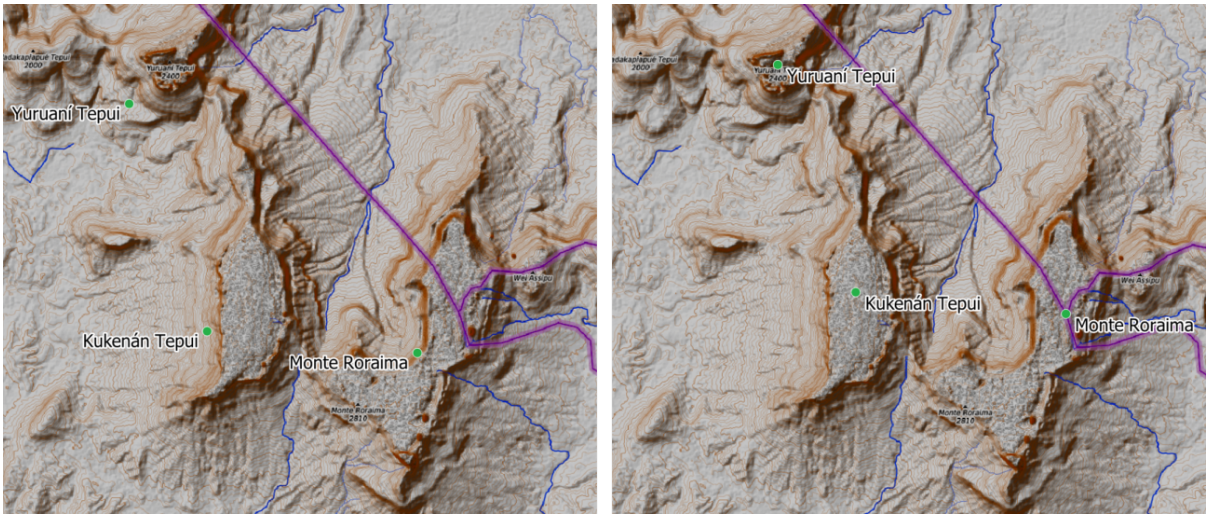


図 27.50: アフィン変換（平行移動）する前（左）と後（右）のベクタポイントレイヤ（緑色の点）。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

平行移動（*translate*）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
平行移動（X 軸）	DELTA_X	[数値] デフォルト：0	X 軸方向に適用する移動量
平行移動（Y 軸）	DELTA_Y	[数値] デフォルト：0	Y 軸方向に適用する移動量
平行移動（Z 軸）	DELTA_Z	[数値] デフォルト：0	Z 軸方向に適用する移動量
平行移動（M 値）	DELTA_M	[数値] デフォルト：0	M 値に適用するオフセット値
縮尺（X 軸）	SCALE_X	[数値] デフォルト：1	X 軸方向に適用するスケーリング値（拡大または縮小）

次のページに続く

表 27.133 – 前のページからの続き

ラベル	名前	データ型	説明
縮尺（Y 軸）	SCALE_Y	[数値] デフォルト： 1	Y 軸方向に適用するスケーリング値（拡大または縮小）
縮尺（Z 軸）	SCALE_Z	[数値] デフォルト： 1	Z 軸方向に適用するスケーリング値（拡大または縮小）
縮尺（M 値）	SCALE_M	[数値] デフォルト： 1	M 値に適用するスケーリング値（拡大または縮小）
Z 軸まわりの回転（度、反時計回り）	ROTATION_Z	[数値] デフォルト： 0	度単位で表した回転角度
変形済みシェープ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
変形済みシェープ	OUTPUT	[入力レイヤと同じ]	（変換された）出力ベクタレイヤ

Python コード

Algorithm ID: native:affinetransform

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

集計

ベクタレイヤまたはテーブルレイヤを入力とし、 式によるグループ化 に基づいて地物を集計した新しいレイヤを作成します。

グループ化 の式で同じ値を返す地物は一つのグループにまとめられます。

式によるグループ化 パラメータに定数値、例えば NULL を使用することで、全てのソース地物を一つにグループ化することができます。

配列関数を使用して、複数のフィールドに基づいたグループ化もできます。例： Array("Field1", "Field2")

（もしあれば）ジオメトリはグループごとに 1 つのマルチパートジオメトリにまとめられます。出力属性は、指定された各集計定義に応じて計算されます。

このアルゴリズムでは、QGIS 式エンジンのデフォルトの **集約関数** を使用できます。

参考:

シングルパートをマルチパートに集約 、 **融合（*dissolve*）**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
式によるグループ化（すべての地物をグループ化する場合は NULL）	GROUP_BY	[テーブルのフィールド：任意] デフォルト：'NULL'	グループ化するフィールドを選択します。 *NULL* の場合、すべての地物が一つにグループ化されます。

次のページに続く

表 27.135 – 前のページからの続き

ラベル	名前	データ型	説明
式によるグループ化（集約）	AGGREGATES	[リスト]	出力レイヤのフィールド定義のリスト。フィールド定義の例は、次のような形式です： *{'aggregate': 'sum', 'delimiter': ',', 'input': ' $area', 'length': 10, 'name': 'totarea', 'precision': 0, 'type': 6}* デフォルトでは、このリストには入力レイヤの全てのフィールドが含まれます。GUI 版では、フィールドやその定義の編集ができ、これに加えて以下の操作ができます： •  ボタンをクリックして、新しいフィールドを追加する •  ボタンをクリックして、選択したフィールドを削除する •  と  でフィールドの順番を並べ替える •  ボタンをクリックして、デフォルト（入力レイヤのフィールド）にリセットする 情報を検索したいフィールドそれぞれについて、以下を定義する必要があります： **ソースの式 [式] (input)** 入力レイヤのフィールドまたは式 **集計関数 [列挙型] (aggregate)** 入力式に対して使用して集計値を得るための **関数** デフォルト：*concatenate*（文字列型の場合）、*sum*（数値型の場合） **区切り文字 [文字列] (delimiter)** 連結の場合など、集計値を区切るためのテキスト文字列。 デフォルト：, **名前 [文字列] (name)** 出力レイヤでの集約フィールドの名前。デフォルトは入力フィールドの名前のまま **型 [列挙型] (type)** 出力フィールドのデータ型。次のいずれかです： • 1 --- ブール値（boolean） • 2 --- 32bit 整数値（integer32） • 4 --- 64bit 整数値（integer64） • 6 --- 小数点付き数値（double） • 10 --- テキスト（string） • 14 --- 日付（Date） • 16 --- 日付時刻（DateTime） **長さ [数値] (length)** 出力フィールドの長さ

表 27.135 – 前のページからの続き

ラベル	名前	データ型	説明
テンプレートレイヤから属性を読み込む	GUI 版のみ	[ベクタ：任意]	別のレイヤからフィールドを読み込み、集約計算に使用することができます
集計出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（集約値の）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
集計出力	OUTPUT	[入力レイヤと同じ]	集約値を持つマルチジオメトリのベクタレイヤ

Python コード

Algorithm ID: `native:aggregate`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

トポロジカル境界

入力ジオメトリの組み合わせ境界の閉包（すなわちジオメトリのトポロジカル境界）を返します。

ポリゴンレイヤとラインレイヤにのみ使用可能です。

ポリゴンジオメトリ の場合は、境界はポリゴンのリングを構成するすべてのラインからなります。

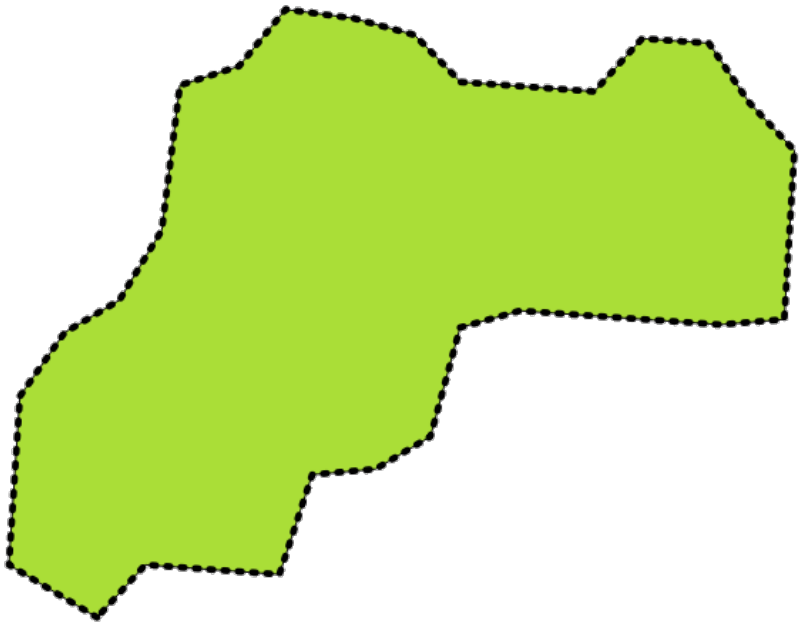


図 27.51: ソースポリゴンレイヤの境界（黒色の破線）

ラインジオメトリ の場合は、境界はラインの端点です。

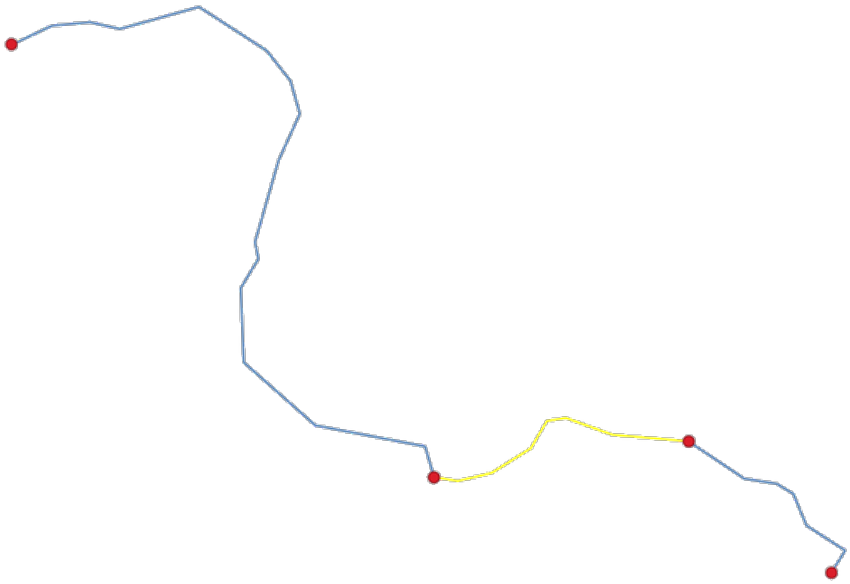


図 27.52: ラインの境界レイヤ（赤色の点）。黄色は選択されたライン地物

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
トポロジカル境界	OUTPUT	[ベクタ：ポイント、ライン] デフォルト：[一時レイヤを作成]	（トポロジカル境界の）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
トポロジカル境界	OUTPUT	[ベクタ：ポイント、ライン]	入力レイヤのトポロジカル境界（入力がラインの場合はポイント、ポリゴンの場合はライン）

Python コード

Algorithm ID: native:boundary

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

BBox の出力

入力レイヤ内の各地物のバウンディングボックス（エンベロープ）を計算します。ポリゴンおよびライン ジオメトリがサポートされています。

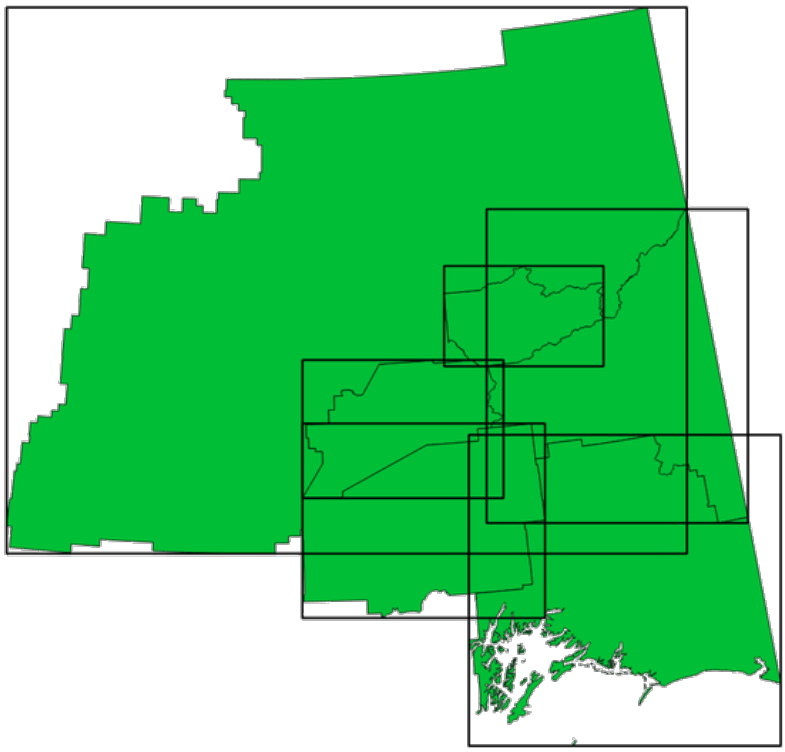


図 27.53: 黒色のラインは各ポリゴン地物のバウンディングボックスを表す

ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

最小境界ジオメトリ

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
境界	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（バウンディングボックスの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
境界	OUTPUT	[ベクタ：ポリゴン]	入力レイヤのバウンディングボックス

Python コード

Algorithm ID: native:boundingboxes

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

バッファ（buffer）

固定又はデータで定義する距離を使って、入力レイヤの全地物のバッファ領域を計算します。

入力レイヤがポリゴンの場合は、負の距離を使用することもできます。この場合、バッファはポリゴンよりも小さくなります（セットバック）。

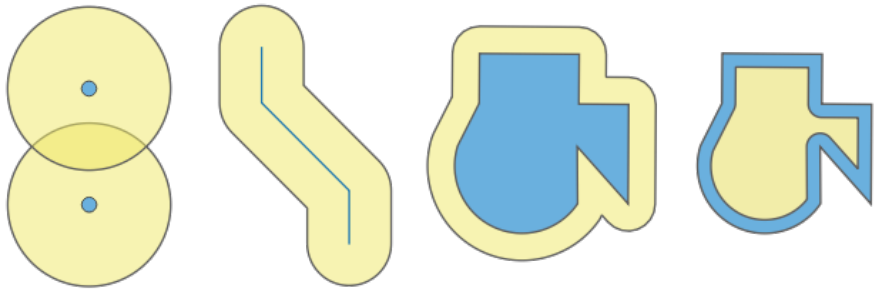


図 27.54: バッファ（黄色）。正のバッファを持ったポイント、ライン、ポリゴン及び負のバッファを持ったポリゴン

ポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： **ベクタ　空間演算ツール**

参考:

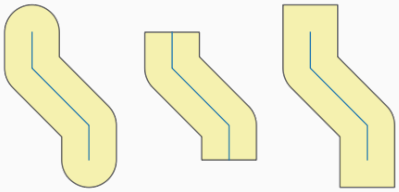
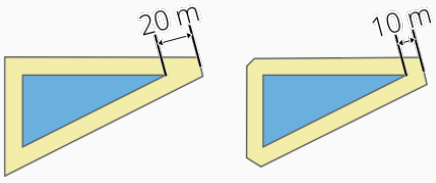
可変距離バッファ 、 **多重リングバッファ** 、 **可変幅バッファ（*M* 値使用）**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
距離	DISTANCE	[数値] デフォルト： 10.0	（各地物の境界から計算された）バッファ距離。右側にある「データによって定義された上書き」ボタンを使用すると、バッファ距離を計算するためのフィールドを選択できます。これにより、各地物で異なるバッファ距離を設定できます（ **可変距離バッファ** も参照）。
セグメント	SEGMENTS	[数値] デフォルト： 5	丸みを帯びたオフセットの四分円を近似するために使用するセグメントの数を制御します

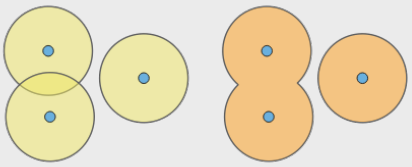
次のページに続く

表 27.136 – 前のページからの続き

ラベル	名前	データ型	説明
線端スタイル	END_CAP_STYLE	[列挙型] デフォルト： 0	ラインの端がバッファでどのように処理されるかを制御します。次のいずれかです： • 0 --- Round • 1 --- Flat • 2 --- Square  図 27.55: Round、flat、square の線端スタイル
継ぎ目スタイル	JOIN_STYLE	[列挙型] デフォルト： 0	ラインの角をオフセットする場合に使用する継ぎ目スタイルを round、miter、bevel から指定します。オプションは次のとおり： • 0 --- Round • 1 --- Miter • 2 --- Bevel  図 27.56: round、miter、bevel のつなぎ目スタイル
miter 制限	MITER_LIMIT	[数値] デフォルト： 2.0	miter 継ぎ目を作成する際に使用するオフセットジオメトリからの最大距離を、オフセット距離の係数として設定します（miter 継ぎ目スタイルにのみ適用されます）。最小値: 1.0  図 27.57: 10m バッファで制限 2 の場合と、10m バッファで制限 1 の場合

次のページに続く

表 27.136 – 前のページからの続き

ラベル	名前	データ型	説明
結果を融合する	DISSOLVE	[ブール値] デフォルト：False	最終バッファを融合します。True (チェックされた) の場合、重なり合っているバッファは 1 つのマルチパート地物に融合（結合）されます。  図 27.58: 標準（3 つのシングルパート地物 - 左)、融合（2 つの部分を持ったひとつのマルチパート地物 - 右）
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（バッファの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	（バッファの）出力ポリゴンレイヤ

Python コード

Algorithm ID: native:buffer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

重心

入力レイヤのジオメトリの重心を表すポイントレイヤを新たに作成します。

重心は、地物の（すべての部分の）数学的な重心を表す単一の点であるため、地物の境界の外側にあることもあります。地物の各パートで重心を作成することもできます。

出力レイヤのポイントの属性は、元のレイヤの属性と同じです。

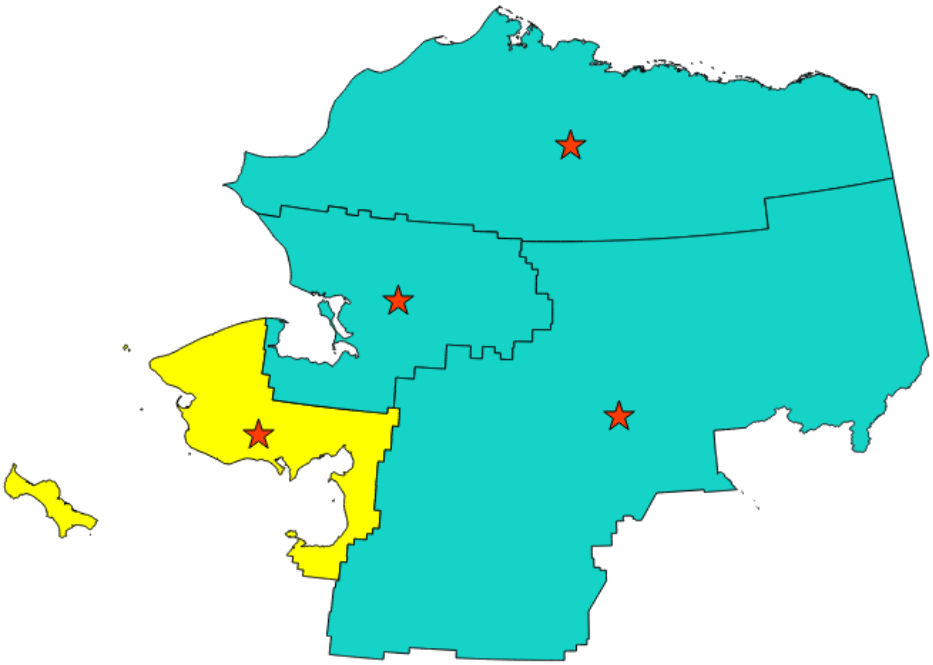


図 27.59: 赤色の星印は入力レイヤの各地物の重心を表す

ライン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： **ベクタ　ジオメトリツール**

参考:

内部保証点（*point on surface*）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
各パートに重心を作成	ALL_PARTS	[ブール値] デフォルト：False	True（チェックされている）場合には、マルチパートジオメトリの各部分について重心を作成します
重心	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	（重心の）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
重心	OUTPUT	[ベクタ：ポイント]	（重心の）出力ポイントベクタレイヤ

Python コード

Algorithm ID: native:centroids

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

有効性チェック

ベクタレイヤのジオメトリに対して有効性チェックを実行します。

ジオメトリは３つのグループ（有効、不正、エラー）に分類され、以下のように、各グループについての地物を持ったベクタレイヤを生成します：

- **有効な出力** レイヤには、有効な地物（トポロジカルエラーを除く）のみが含まれます。
- **不正な出力** レイヤには、このアルゴリズムで検出された不正なジオメトリの地物がすべて含まれます。
- **エラー出力** レイヤはポイントレイヤで、どこに地物の不正があったかを示します。

生成されるレイヤの属性テーブルには、いくつか追加情報が含まれます（**エラー** レイヤには "message" が、**不正な** レイヤには "FID" と "_errors" が、 **有効な** レイヤには "FID" のみが追加されます）。

生成される各ベクタレイヤの属性テーブルには、いくつか追加情報が含まれます（見つかったエラーの数や、エラーの種類）

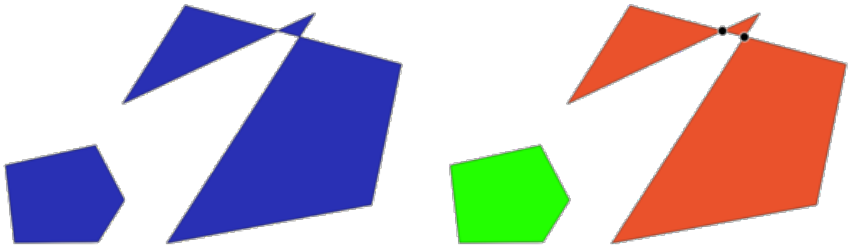


図 27.60: 左：入力レイヤ　右：有効なレイヤ（緑）と不正なレイヤ（オレンジ）

デフォルトメニュー ： **ベクタ　ジオメトリツール**

参考:

ジオメトリを修復 およびコアプラグインの **ジオメトリチェッカープラグイン**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT_LAYER	[ベクタ：任意]	入力ベクタレイヤ
方法	METHOD	[列挙型] デフォルト： 2	有効性チェックに使用する手法。オプションは以下のとおり： • 0: デジタイジング設定の選択 • 1: QGIS • 2: GEOS
自己交差するリングを無視	IGNORE_RING_SELF	[ブール値] デフォルト： False	有効性チェックの際に自己交差するリングを無視します

次のページに続く

表 27.137 – 前のページからの続き

ラベル	名前	データ型	説明
有効なジオメトリの出力レイヤ	VALID_OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	ソースレイヤのうち、有効なジオメトリを持つ地物のコピーが含まれるベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
不正なジオメトリの出力レイヤ	INVALID_OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	ソースレイヤのうち、不正なジオメトリを持つ地物のコピーが含まれるベクタレイヤを指定します。 _errors フィールドには見つかったエラーの概要が含まれます。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
エラー部分の出力レイヤ	ERROR_OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	ジオメトリ有効性の問題が検出された正確な位置を表すポイントレイヤを指定します。 message フィールドには見つかったエラーの説明が含まれます。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
エラーの数	ERROR_COUNT	[数値]	エラーが発生したジオメトリの数
エラー部分の出力レイヤ	ERROR_OUTPUT	[ベクタ：ポイント]	ジオメトリ有効性の問題が検出された正確な位置を表すポイントレイヤ。message フィールドには見つかったエラーの説明が含まれます。
不正な地物の数	INVALID_COUNT	[数値]	不正なジオメトリの数
不正なジオメトリの出力レイヤ	INVALID_OUTPUT	[入力レイヤと同じ]	ソースレイヤのうち、不正なジオメトリを持つ地物のコピーが含まれるベクタレイヤ。 _errors フィールドには見つかったエラーの概要が含まれます。
正しい地物数	VALID_COUNT	[数値]	有効なジオメトリの数
有効なジオメトリの出力レイヤ	VALID_OUTPUT	[入力レイヤと同じ]	ソースレイヤのうち、有効なジオメトリを持つ地物のコピーが含まれるベクタレイヤ。

Python コード

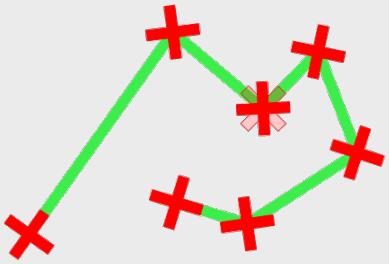
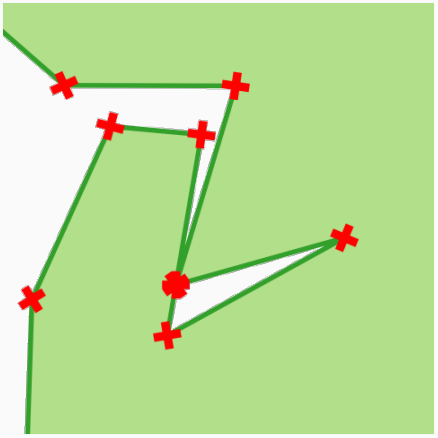
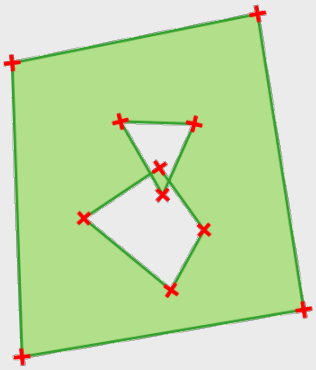
Algorithm ID: `qgis:checkvalidity`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

エラーメッセージの種類と意味

表 27.139: GEOS の方法を使用する場合に現れるエラーメッセージ

エラーメッセージ	説明	例
重複点	このエラーは、頂点が重なっている場合に発生します。	
リングの自己交差	このエラーは、ジオメトリが自身と交わってリングを生じる場合に発生します。	
自己交差	このエラーは、ジオメトリが自身と交わる場合に発生します。	
トポロジ検証エラー		
孔が外側に飛び出ています		
孔が入れ子になっている		
内部が繋がっていません		

次のページに続く

表 27.139 – 前のページからの続き

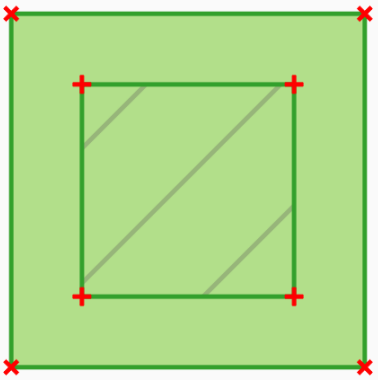
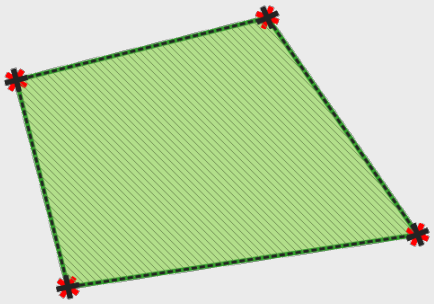
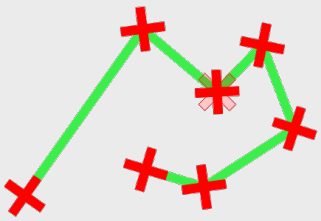
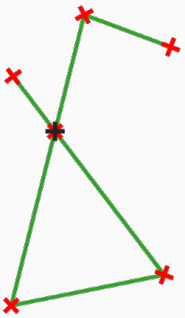
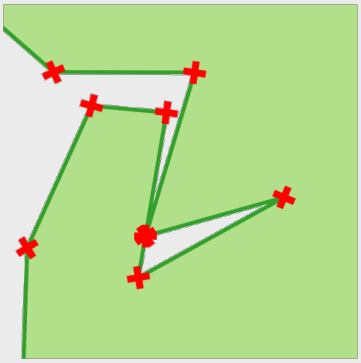
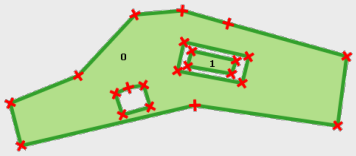
エラーメッセージ	説明	例
外殻がネストしています	このエラーは、あるポリゴンジオメトリが別のポリゴンジオメトリ上にある場合に発生します。	
リングの重複	このエラーは、ポリゴンジオメトリの 2 つのリング（外部または内部）が同一である場合に発生します。	
頂点の数が少なすぎます		
不正な座標	ポイントジオメトリの場合、このエラーはジオメトリが適切な座標ペアを持っていない場合に発生します。座標ペアには、経度値と緯度値がこの順番で含まれていない場合などです。	
リングが閉じていません		

表 27.140: QGIS の方法を使用する場合に現れるエラーメッセージ

エラーメッセージ	説明	例
ポリゴン %3 のリング %2 の線分 %1 は、ポリゴン %6 のリング %5 の線分 %4 と点 %7 で交差します		
%1 のリングは 4 個未満の点で構成されています		
%1 のリングは閉じていません		

次のページに続く

表 27.140 – 前のページからの続き

エラーメッセージ	説明	例
%1 のラインは 2 点未満の点で構成されています		
%1 の線は、点 %2 に重複ノードを %n 個持っています	このエラーは、ラインの連続する頂点が同じ座標である場合に発生します。	
線 %3 の線分 %1 と %2 は、 %4 で交差しています	このエラーは、ラインが自己交差する（ラインの 2 つのセグメントが互いに交わる）場合に発生します。	
リングの自己交差	このエラーは、ポリゴンジオメトリの外部境界または内部（島）リングが自分自身と交わる場合に発生します。	
ポリゴン%2 のリング%1 は外部リング内にありません		
ポリゴン %1 がポリゴン %2 の内側にあります	このエラーは、マルチポリゴンジオメトリのパートがマルチポリゴンジオメトリの孔の内部にある場合に発生します。	

シングルパートをマルチパートに集約

ベクタレイヤを入力として、そのジオメトリを新しいマルチパートのジオメトリに集約します。

1 つ以上の属性を指定して同じクラス（指定された属性について同じ値を持つ）に属するジオメトリだけを集約することもできますが、全てのジオメトリをマルチパートジオメトリに集約することもできます。

たとえジオメトリがただ一つでも、すべての出力ジオメトリはマルチパートのジオメトリに変換されます。このアルゴリズムは、重なり合ったジオメトリをディゾルブ（融合）しません。各ジオメトリパートの形状は変更されることなく、一つのマルチパートジオメトリに集約されます。

別の選択肢として、「マルチパートに強制変換」や「集計」アルゴリズムも参照してください。

デフォルトメニュー ： ベクタ　 ジオメトリツール

参考:

集計 、 マルチパートに強制変換 、 融合（*dissolve*）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
属性（フィールド）	FIELD	[テーブルのフィールド：任意] [リスト]	ジオメトリを集約するための属性を 1 つ以上選択します
出力レイヤ	OUTPUT	[入力レイヤと同じ]	集約されたジオメトリのベクタレイヤ

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	集約されたジオメトリの出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

Python コード

Algorithm ID: `native:collect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

凹包（アルファシェイプ）

入力ポイントレイヤの地物の凹包を計算します。

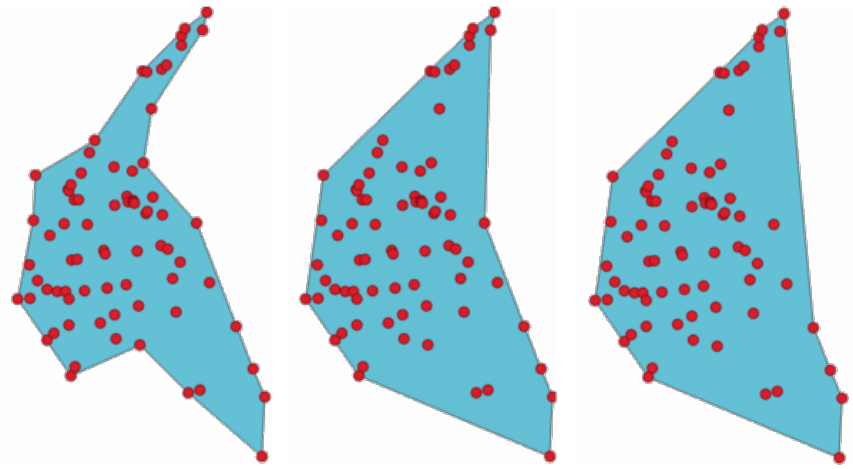


図 27.61: さまざまな閾値（0.3、0.6、0.9）による凹包

参考:

凸包（*convex hull*） 、 **凹包（*k* 近傍法）**

パラメータ

ラベル	名前	データ型	説明
入力点のレイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
閾値	ALPHA	[数値] デフォルト： 0.3	0（凹包の限界）から 1（凸包）までの数値
穴を許す	HOLES	[ブール値] デフォルト： True	結果の凹包に穴があることを許すかどうかを選択します

次のページに続く

表 27.141 – 前のページからの続き

ラベル	名前	データ型	説明
マルチパートはシングルパートに分割する	NO_MULTIGEOMETRY	[ブール値] デフォルト： True	マルチパートジオメトリではなく、シングルパートジオメトリの結果が欲しい場合にはチェックしてください
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	出力ベクタレイヤ

Python コード

Algorithm ID: `qgis:concavehull`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

凹包（k 近傍法）

点の集合から凹包ポリゴンを生成します。入力レイヤがラインレイヤまたはポリゴンレイヤの場合には、その頂点が使用されます。

考慮する隣接点の数によって、出力ポリゴンの凹度が決まります。数値が小さいほどポイントを追従する凹包になり、反対に数値が大きいほどポリゴンは滑らかな形状になります。隣接点の最小値は 3 です。点の数かそれ以上の値の場合には、凸包となります。

属性を選択した場合、このアルゴリズムは入力レイヤの地物をその属性のユニーク値でグループ化し、各グループに対して個別のポリゴンを出力レイヤに生成します。

参考:

凹包（アルファシェイプ）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
隣接点の数（少ないほど凹形になり、大きいほど滑らかになります）	KNEIGHBORS	[数値] デフォルト： 3	出力ポリゴンの凹度を決める値。数値が小さいほどポイントを追従する凹包となり、反対に数値が大きいほどポリゴンは凸包に似てくる（値がポイント地物数以上の場合には、凸包となる）。最小値は 3
属性 オプション	FIELD	[テーブルのフィールド：任意] デフォルト： なし	指定した場合、（ユニーク値を使用して地物選択することで）このフィールドのユニーク値ごとに 1 つの凹包ポリゴンが生成される
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	出力ベクタレイヤ

Python コード

Algorithm ID: `qgis:knearestconcavehull`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリタイプの変換

既存のレイヤに基づいて、異なるジオメトリタイプの新しいレイヤを作成します。

出力レイヤの属性テーブルは、入力レイヤのものと同じです。

すべての変換が可能なわけではありません。例えば、ラインレイヤはポイントレイヤへ変換できますが、ポイントレイヤはラインレイヤに変換できません。

参考:

ポリゴン化（*Polygonize*） 、 **線をポリゴンに変換（*lines to polygons*）** 、 **ポリゴンを線に変換** 、 **点を線に変換**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
新しいジオメトリタイプ	TYPE	[列挙型] デフォルト：0	出力レイヤの地物のジオメトリタイプ。次のいずれかです： • 0 --- 重心 • 1 --- 構成点（ノード） • 2 --- ラインストリング • 3 --- マルチラインストリング • 4 --- ポリゴン（Polygons）
出力レイヤ	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：任意]	出力ベクタレイヤ。ジオメトリタイプはパラメータによります

Python コード

Algorithm ID: `qgis:convertgeometrytype`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

曲線ジオメトリに変換

あるジオメトリを等価な曲線ジオメトリに変換します。

もともと曲線のジオメトリは変換されず、そのままです。

ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ラインまたはポリゴン]	入力ベクタレイヤ
最大許容距離	DISTANCE	[数値] デフォルト：0.000001	元の頂点の位置と、これに対応する変換後の曲線ジオメトリ上の点との間に許容される最大距離
最大許容角（度）	ANGLE	[数値] デフォルト：0.000001	置き換え候補の円弧上にすべての点が規則的に配置されている場合、そのセグメントは円弧に置き換えるのに適していると判断されます。このパラメータは、点が規則的な間隔であるかをテストする際に許容される最大角度偏差（度）を指定します。値は 0 から 45 ° の範囲です
曲線	OUTPUT	[ベクタ：複合カーブまたはカーブポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
曲線	OUTPUT	[ベクタ：複合カーブまたはカーブポリゴン]	曲線ジオメトリの出力ベクタレイヤ

Python コード

Algorithm ID: `native:converttocurves`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

凸包（convex hull）

入力レイヤの各地物の凸包を計算します。

レイヤ全体またはグループ化された地物のサブセットを覆う凸包の計算については「最小境界ジオメトリ」アルゴリズムを参照してください。

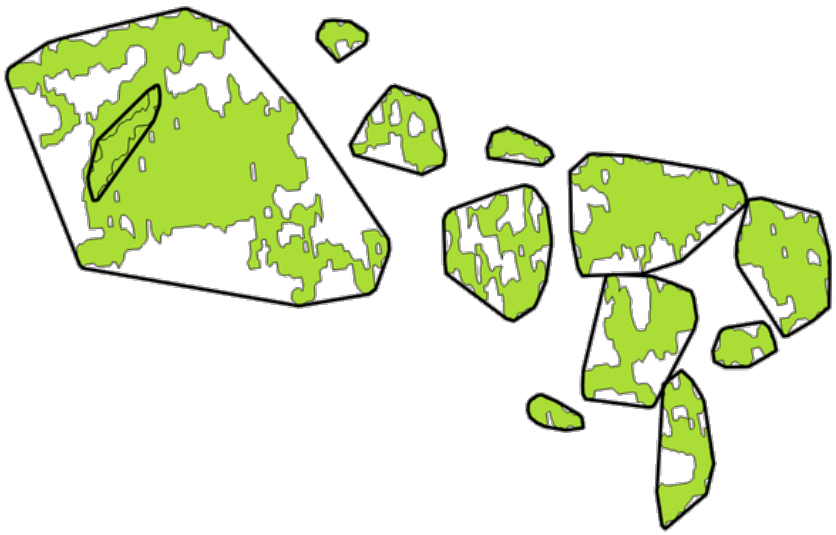


図 27.62: 黒線はレイヤの各地物の凸包を示す

ポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： **ベクタ　空間演算ツール**

参考:

最小境界ジオメトリ 、 **凹包（アルファシェイプ）**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	（凸包の）出力ベクタレイヤ

Python コード

Algorithm ID: native:convexhull

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

範囲の矩形レイヤ

入力レイヤの範囲と同一のジオメトリの単一の矩形地物を含んだ、新しいベクタレイヤを作成します。

グラフィカルモデラー内で使用して、(xmin, xmax, ymin, ymax) 形式のリテラル範囲をレイヤに変換し、レイヤ形式での入力を必要とする他のアルゴリズムに使用することができます。

参考:

点からレイヤを作成

パラメータ

ラベル	名前	データ型	説明
領域	INPUT	[範囲]	入力の範囲 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • xmin, xmax, ymin, ymax として座標を入力
領域	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
領域	OUTPUT	[ベクタ：ポリゴン]	（範囲の）出力ベクタレイヤ

Python コード

アルゴリズム ID: native:extenttolayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

点からレイヤを作成

ポイントを表すパラメータと同一のジオメトリの単一の地物を含んだ、新しいベクタレイヤを作成します。グラフィカルモデラー内で使用して、点をポイントレイヤに変換し、レイヤ形式での入力を必要とするアルゴリズムに使用することができます。

参考:

範囲の矩形レイヤ

パラメータ

ラベル	名前	データ型	説明
点（Point）	INPUT	[座標]	CRS 情報を含んだ点の入力（例：397254,6214446 [EPSG:32632]） CRS が指定されない場合、プロジェクトの CRS が使用されます。 ポイントは、マップキャンバス上をクリックすることでも指定できます。
点（Point）	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
点（Point）	OUTPUT	[ベクタ：ポイント]	入力点を含んだ出力ポイントベクタレイヤ

Python コード

Algorithm ID: native:pointtolayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ウェッジバッファを作成

入力ポイントからくさび形のバッファを作成します。

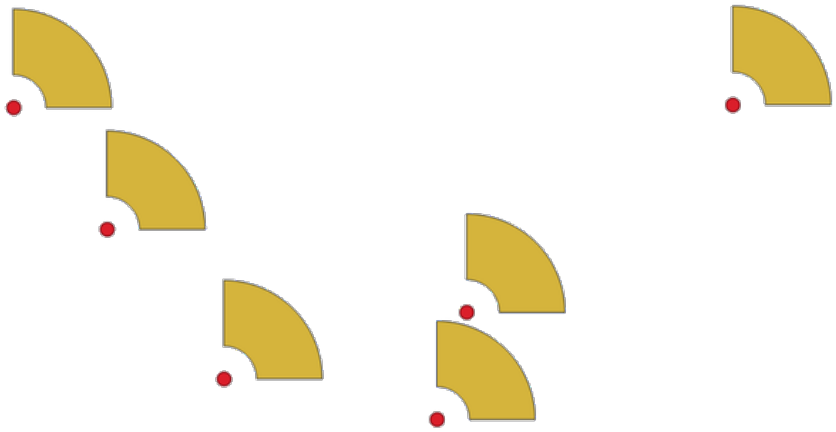


図 27.63: ウェッジバッファ

このアルゴリズムのネイティブ出力は CurvePolygon ジオメトリですが、出力フォーマットによっては自動的にセグメント化されてポリゴンとなる場合があります。

参考:

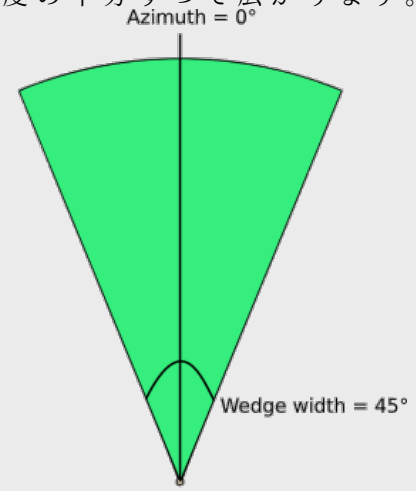
バッファ（*buffer*）、**可変幅バッファ（*M* 値使用）**、**テイパードバッファ**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
方位角（単位は度、北がゼロの時計回り）	AZIMUTH	[数値] デフォルト： 0.0	ウェッジの中央の方位角（度単位）

次のページに続く

表 27.143 – 前のページからの続き

ラベル	名前	データ型	説明
ウェッジ幅（単位は度）	WIDTH	[数値] デフォルト：45.0	バッファの幅（度単位）。ウェッジは方位角の方向の両側にこの角度の半分ずつで広がります。  図 27.64: ウェッジバッファの方位角と幅の値
外径	OUTER_RADIUS	[数値] デフォルト：1.0	ウェッジの外側の **サイズ**（長さ）：この「サイズ」は、原点からくさび形の端までの長さを意味します。
内径 オプション	INNER_RADIUS	[数値] デフォルト：0.0	内側の半径の値。0 の場合、ウェッジは原点から始まります。
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	（ウェッジバッファの）出力ベクタレイヤ

Python コード

Algorithm ID: `native:wedgebuffers`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ドロネー三角分割

入力ポイントレイヤに対応するドロネー三角分割のポリゴンレイヤを作成します。

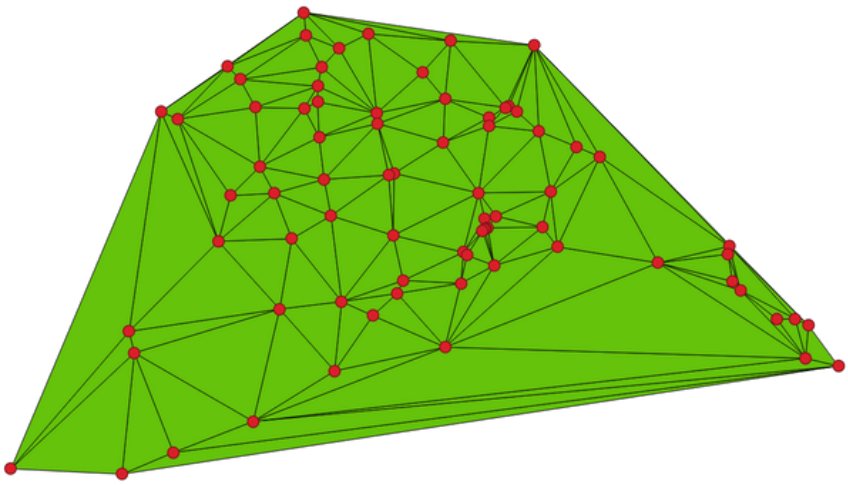


図 27.65: ポイントのドロネー三角分割

デフォルトメニュー ： ベクタ　 ジオメトリツール

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
ドロネー三角分割	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ドロネー三角分割	OUTPUT	[ベクタ：ポリゴン]	（ドロネー三角分割の）出力ベクタレイヤ

Python コード

Algorithm ID: `qgis:delaunaytriangulation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

孔（hole）の削除

ポリゴンレイヤを入力として、ポリゴンの孔を削除します。このアルゴリズムは新しいベクタレイヤを作成しますが、孔のあるポリゴンは外側リングのみのポリゴンに置き換えられます。属性は変更されません。

オプションの最小面積パラメータを使用すると、指定された閾値よりも面積が小さい孔のみを除去できます。このパラメータを `0.0` のままにすると、すべての孔を削除します。

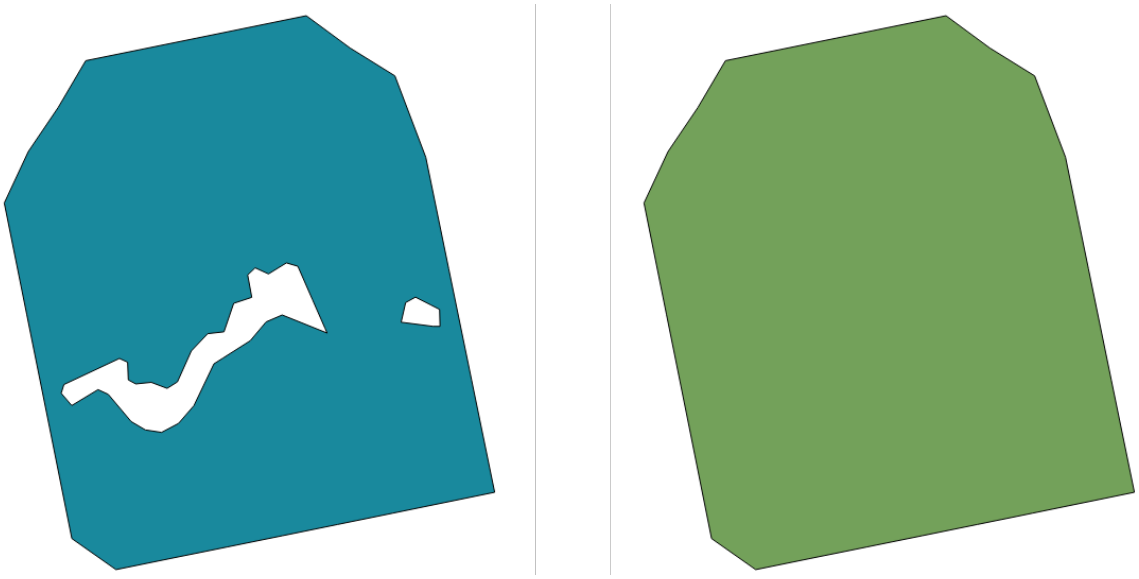


図 27.66: ポリゴンの孔の削除前と削除後

ポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	`INPUT`	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
この面積より小さな孔を削除 オプション	`MIN_AREA`	[数値] デフォルト： 0.0	この閾値よりも小さい面積の孔のみを削除します。この値を `0.0` とすると、 **すべて** の孔を削除します。
クリーニング済み出力	`OUTPUT`	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
クリーニング済み出力	OUTPUT	[入力レイヤと同じ]	（孔が削除された）出力ベクタレイヤ

Python コード

Algorithm ID: native:deleteholes

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

頂点の高密度化（個数ベース）

ポリゴンまたはラインレイヤを入力として、元の頂点よりも多数の頂点を持ったジオメトリのレイヤを新たに作成します。

ジオメトリが Z 座標や M 値を持つ場合には、追加された頂点には値が線形補間されます。

新たに各セグメントに追加する頂点数は、入力パラメータで指定します。

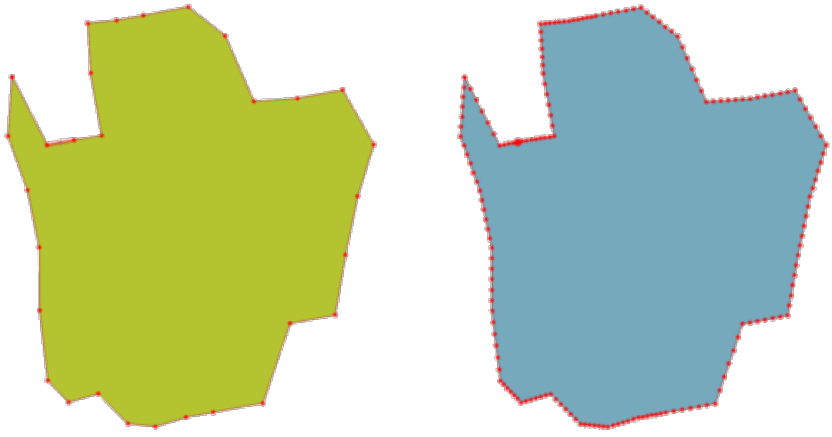


図 27.67: 頂点の高密度化の実行前後。赤色の点は頂点を表す

ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： **ベクタ　ジオメトリツール**

参考:

間隔による高密度化

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
地物あたりの追加頂点数	VERTICES	[数値] デフォルト： 1	各セグメントに追加する頂点の数
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（頂点が高密度化された）出力ベクタレイヤ

Python コード

Algorithm ID: native:densifygeometries

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

間隔による高密度化

ポリゴンまたはラインレイヤを入力として、元の頂点よりも多数の頂点を持ったジオメトリのレイヤを新たに作成します。

2 つの頂点間の最大距離が指定された距離を超えないよう、余分な頂点を各セグメントの間に規則的に追加配置することによって、ジオメトリの頂点を高密度化します。

ジオメトリが Z 座標や M 値を持つ場合には、追加された頂点には値が線形補間されます。

例

間隔の上限に 3 を指定した場合、セグメント `[0 0] -> [10 0]` は、 `[0 0] -> [2.5 0] -> [5 0] -> [7.5 0] -> [10 0]` のように変わります。これは、セグメント上に 3 つの追加の頂点が必要で、これらを 2.5 間隔で配置するとセグメント上で等間隔に配置できるためです。

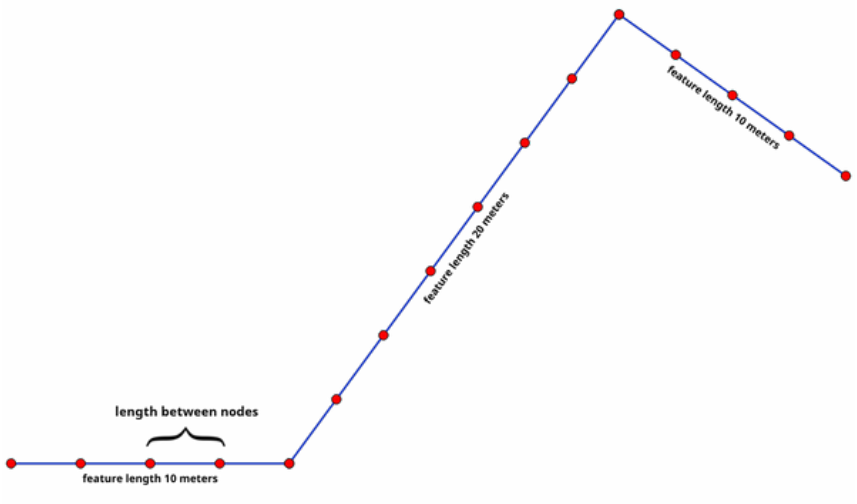


図 27.68: 指定した間隔によるジオメトリの高密度化

ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

頂点の高密度化（個数ベース）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
頂点の間隔の上限	INTERVAL	[数値] デフォルト： 1.0	2 つの連続する頂点間の最大距離
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（頂点が高密度化された）出力ベクタレイヤ

Python コード

Algorithm ID: native:densifygeometriesgivenaninterval

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

融合（dissolve）

ベクタレイヤを入力として、その地物を結合して新しい地物にします。1 つ以上の属性を指定して、同じクラスに属する（指定した属性に同じ値を持つ）地物どうしを融合することができます。また、すべての地物を単一の地物へと融合させることもできます。

すべての出力ジオメトリはマルチジオメトリに変換されます。入力がポリゴンレイヤの場合、融合される隣接ポリゴンの共通の境界線は消去されます。オプションの「接していない地物は融合しない」を有効にすると、重なったり接触したりしない地物や部分は、（ひとつのマルチパート地物の部分ではなく）別々の地物としてエクスポートされます。

結果の属性テーブルは、入力レイヤと同じフィールドを持ちます。出力レイヤのフィールドの値は、処理に入力された最初の地物の値です。

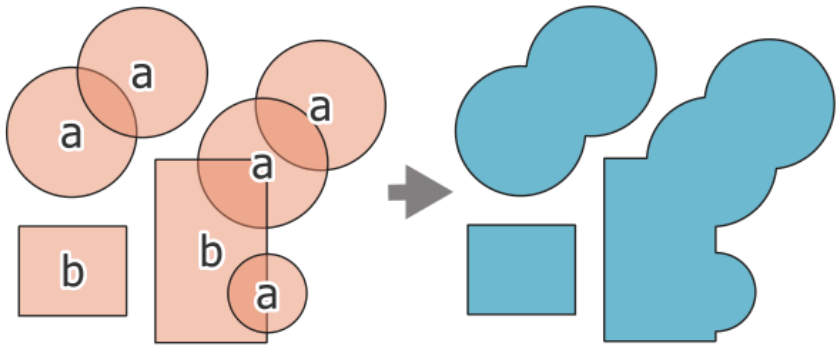


図 27.69: ひとつのレイヤをひとつのマルチパート地物に融合する

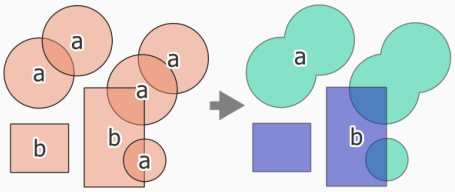
デフォルトメニュー ： ベクタ　 空間演算ツール

参考:

集計 、 シングルパートをマルチパートに集約

パラメータ

基本パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
基準となる属性(複数可) オプション	FIELD	[テーブルのフィールド：任意] [リスト] デフォルト： []	選択したフィールド（複数可）に同じ値を持つ地物を、ジオメトリが結合した単一の地物に置き換えます。 フィールドを指定しない場合、すべての地物を融合させ、結果は単一の（マルチパート）地物となります。  図 27.70: ポリゴンレイヤを共通の属性で融合する（2 つのマルチパート地物）
融合ポリゴンの出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	データ型	説明
接していない地物は融合しない NEW in 3.26	SEPARATE_DISJOIN	[ブール値] デフォルト：False	融合された地物の部分は、（マルチパート地物の部分ではなく）別々の地物としてエクスポートされます。

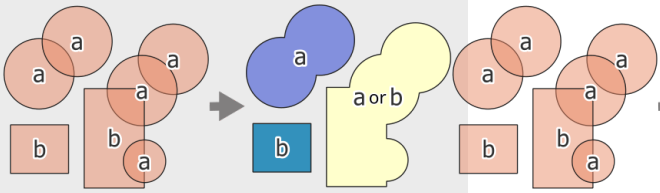


図 27.71: ソース（左）、全てを融合（3つの別々の地物 - 右）

図 27.72: ソース（合（5 つの別々の地

出力

ラベル	名前	データ型	説明
融合ポリゴンの出力	OUTPUT	[入力レイヤと同じ]	ジオメトリを融合させた出力ベクタレイヤ

Python コード

Algorithm ID: native:dissolve

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ドレープ（ラスタ値を Z 値に代入）

ラスタレイヤ内のバンドからサンプリングした値を使用して、地物ジオメトリが重なる頂点すべてに Z 値を設定します。ラスタ値は、オプションでプリセット量によりスケーリングできます。

レイヤに Z 値が既にある場合には、新しい値で上書きされます。レイヤが Z 値を持たない場合には、ジオメトリが Z 次元を持つように更新されます。

☑ ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

ドレープ（ラスタ値を *M* 値に代入）、*Z* 値の設定

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
ラスタレイヤ	RASTER	[ラスタ]	Z 値となるラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	Z 値を取得するラスタのバンド
ラスタの nodata 部分もしくは外側にある頂点の値	NODATA	[数値] デフォルト： 0	ラスタ（の有効なピクセル）と交差しない頂点に使用する値
縮尺係数	SCALE	[数値] デフォルト： 1.0	縮尺係数：バンド値にこの値が掛かります。
オフセット NEW in 3.28	OFFSET	[数値] デフォルト： 0.0	オフセット値：「縮尺係数」を適用した後、代数的にバンド値に加算されます。
ドレープ出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（ラスタレイヤから Z 値を設定した）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ドレープ出力	OUTPUT	[入力レイヤと同じ]	ラスタレイヤから Z 値を設定した出力ベクタレイヤ

Python コード

Algorithm ID: `native:setzfromraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

M 値 Z 値の削除

入力ジオメトリから M 値（measure）や Z 値（高度）を削除します。

参考:

M **値の設定** 、 *Z* **値の設定**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	M 値や Z 値を持った入力ベクタレイヤ
M 値の削除	DROP_M_VALUES	[ブール値] デフォルト：False	ジオメトリから M 値を削除します
Z 値の削除	DROP_Z_VALUES	[ブール値] デフォルト：False	ジオメトリから Z 値を削除します
Z 値/M 値の削除	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
Z 値/M 値の削除	OUTPUT	[入力レイヤと同じ]	出力ベクタレイヤ（M 値や Z 次元が削除されていることを除けば入力レイヤと同一）

Python コード

Algorithm ID: native:dropmzvalues

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

選択物の隣接ポリゴンの融合（eliminate）

入力レイヤの選択中のポリゴンと、ある隣接するポリゴンの共通境界を消去して結合します。隣接ポリゴンは、面積が最大のもの、または最小のもの、あるいは削除されるポリゴンと最大の共通境界を共有するもののいずれかです。

このアルゴリズムは通常、スライバポリゴン、すなわち境界が似ているが同一ではないポリゴンの交差計算を行った結果生じる小さなポリゴンを除去するために使います。

デフォルトメニュー ： ベクタ　空間演算ツール

参考:

ジオメトリを修復

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
結合する隣接ポリゴンの基準	MODE	[列挙型] デフォルト： なし	選択したポリゴンを削除するために使用するパラメータを以下から選択します： • 0 --- 最大の面積 • 1 --- 最小の面積 • 2 --- 最大の共通境界
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	出力結果のポリゴンベクタレイヤ

Python コード

Algorithm ID: qgis:eliminateselectedpolygons

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線をセグメントに分解

ラインレイヤを入力として、新しいラインレイヤを作成します。元レイヤの各ラインは、ラインのセグメントを表すラインの集合によって置き換えられます。

結果レイヤの各ラインには始点と終点のみがあり、その中間の頂点はありません。

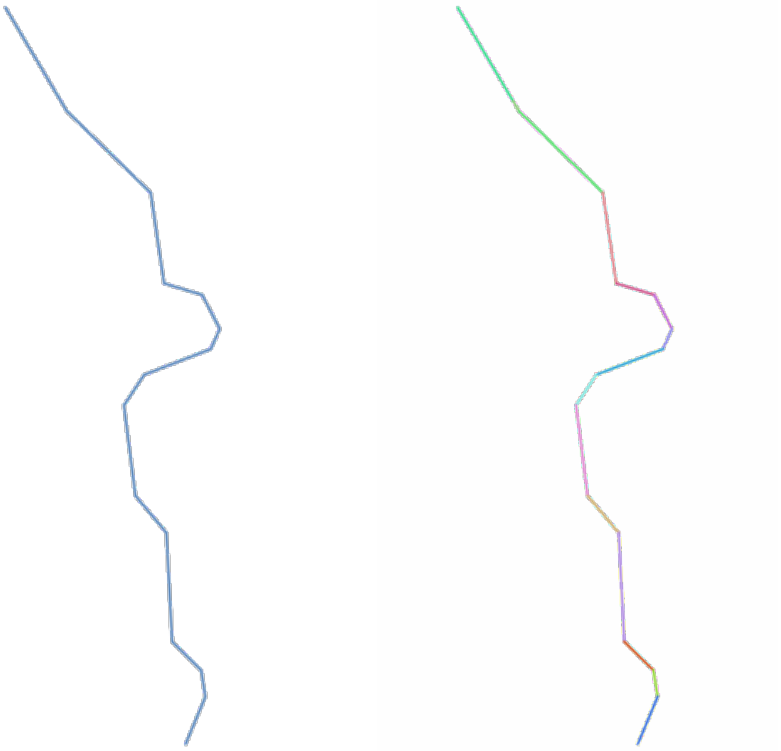


図 27.73: 元のラインレイヤとセグメント分解後

ライン地物の **地物の** *In-place* **編集** が可能です。

参考:

ジオメトリの空間分割 、 **ラインの一部の切り出し**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	入力レイヤの各セグメントを表した地物を持つ出力ラインベクタレイヤ

Python コード

Algorithm ID: native:explodelines

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線の延長

ラインの始点と終点において、指定した量だけラインのジオメトリを延長します。

ラインの最初と最後のセグメントの方向を使用してラインを延長します。

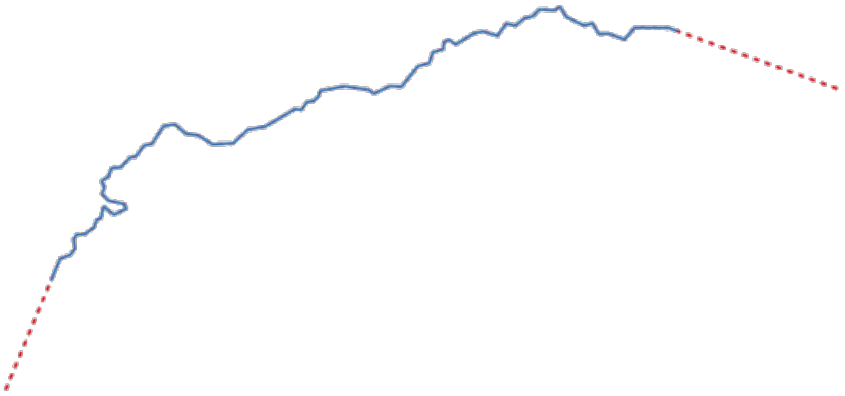


図 27.74: 赤破線は元レイヤの始点と終点の延長を表す

ライン地物の **地物の** *In-place* **編集** が可能です。

参考:

ラインの一部の切り出し

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
始点の延長分	START_DISTANCE	[数値]	ラインの最初のセグメント（始点）を延長する距離
終点の延長分	END_DISTANCE	[数値]	ラインの最後のセグメント（終点）を延長する距離
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	（線が延長された）出力ラインベクタレイヤ

Python コード

Algorithm ID: `native:extendlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

M 値を抽出

ジオメトリから M 値を抽出し、地物の属性テーブルに追加します。

デフォルトでは、ジオメトリの最初の頂点の M 値のみが抜き出されますが、オプションでジオメトリの頂点すべての M 値の統計量（合計、平均、最大・最小など）を計算することもできます。

参考:

Z **値を抽出** 、 *M* **値の設定** 、 *M* **値** *Z* **値の削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ

次のページに続く

表 27.145 – 前のページからの続き

ラベル	名前	データ型	説明
計算する統計値	SUMMARIES	[列挙型] デフォルト： [0]	ジオメトリの M 値に関する統計値。次のうちの 1 つ以上です： • 0 --- 最初（First） • 1 --- 最後（Last） • 2 --- カウント（Count） • 3 --- 合計 • 4 --- 平均 • 5 --- 中央値 • 6 --- 標準偏差（母集団） • 7 --- 最小値 • 8 --- 最大 • 9 --- 範囲（Range） • 10 --- 最稀値（Minority） • 11 --- 最頻値（Majority） • 12 --- 種類（Variety） • 13 --- 第 1 四分位（Q1） • 14 --- 第 3 四分位（Q3） • 15 --- 四分位範囲（IQR）
ラスタ値を収納するカラム名の接頭辞	COLUMN_PREFIX	[文字列] デフォルト： 'm_'	（M 値の）出力カラムの接頭辞
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（M 値を抽出した）出力ベクタレイヤ

Python コード

Algorithm ID: `native:extractmvalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

特定の点を抽出

ベクタレイヤを入力として、入力ジオメトリの特定の頂点を表すポイントレイヤを作成します。

例えば、このアルゴリズムを使用してジオメトリの最初の頂点や最後の頂点を抽出できます。各ポイントに関連付けられる属性は、その頂点が属する地物の属性と同一です。

「頂点インデックス」パラメータには、抽出したい頂点のインデックスを指定するコンマ区切りの文字列を入力します。最初の点はインデックスの 0 に対応し、2 番目の頂点はインデックスの 1、といった具合です。負のインデックスは、ジオメトリの最後の頂点を探すのに使用します。例えば、インデックスの -1 は最後の頂点に対応し、-2 は最後から 2 番目の頂点、などです。

抽出された頂点には、さらに属性が追加されます。これには、頂点位置の指定（例えば 0、-1 など）、元の頂点のインデックス、頂点が含まれていたパートとそのパート内のインデックス（ポリゴンの場合はリングのインデックスも）、元のジオメトリに沿った距離、元のジオメトリの頂点の二等分角度があります。

ライン地物の **地物の** *In-place* **編集** が可能です

参考:

頂点を抽出 、 *M* **値で頂点をフィルタ** 、 *Z* **値で頂点をフィルタ**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
頂点インデックス	VERTICES	[文字列] デフォルト： '0'	抽出したい頂点のインデックスのコンマ区切り文字列
頂点	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
頂点	OUTPUT	[ベクタ：ポイント]	入力レイヤのジオメトリから抽出した特定の頂点の出力（ポイント）ベクタレイヤ

Python コード

Algorithm ID: `native:extractspecificvertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

頂点を抽出

ベクタレイヤを入力として、入力ジオメトリの頂点を表すポイントレイヤを作成します。

各ポイントに関連付けられる属性は、その頂点が属する地物の属性と同一です。

抽出された頂点には、さらに属性が追加されます。これには、頂点のインデックス（0 から始まる）、頂点が含まれていた地物のパートとそのパートのインデックス（ポリゴンの場合はリングのインデックスも）、元のジオメトリに沿った距離、元のジオメトリの頂点の二等分角度があります。

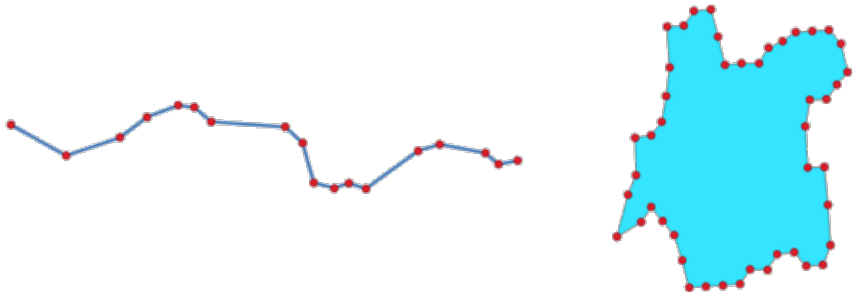


図 27.75: ラインレイヤとポリゴンレイヤから抽出した頂点

ライン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： **ベクタ　ジオメトリツール**

参考:

特定の点を抽出 、 *M* **値で頂点をフィルタ** 、 *Z* **値で頂点をフィルタ**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
頂点	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
頂点	OUTPUT	[ベクタ：ポイント]	入力レイヤのジオメトリから抽出した頂点の出力（ポイント）ベクタレイヤ

Python コード

Algorithm ID: `native:extractvertices`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

Z 値を抽出

ジオメトリから Z 値を抽出し、地物の属性テーブルに追加します。

デフォルトでは、ジオメトリの最初の頂点の Z 値のみが抜き出されますが、オプションでジオメトリの頂点すべての Z 値の統計量（合計、平均、最大・最小など）を計算することもできます。

参考:

M **値を抽出** 、 *Z* **値の設定** 、 *M* **値** *Z* **値の削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ

次のページに続く

表 27.146 – 前のページからの続き

ラベル	名前	データ型	説明
計算する統計値	SUMMARIES	[列挙型] デフォルト： [0]	ジオメトリの Z 値に関する統計値。次のうちの 1 つ以上です： • 0 --- 最初（First） • 1 --- 最後（Last） • 2 --- カウント（Count） • 3 --- 合計 • 4 --- 平均 • 5 --- 中央値 • 6 --- 標準偏差（母集団） • 7 --- 最小値 • 8 --- 最大 • 9 --- 範囲（Range） • 10 --- 最稀値（Minority） • 11 --- 最頻値（Majority） • 12 --- 種類（Variety） • 13 --- 第 1 四分位（Q1） • 14 --- 第 3 四分位（Q3） • 15 --- 四分位範囲（IQR）
ラスタ値を収納するカラム名の接頭辞	COLUMN_PREFIX	[文字列] デフォルト： 'z_'	（Z 値の）出力カラムの接頭辞
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（Z 値を抽出した）出力ベクタレイヤ

Python コード

Algorithm ID: `native:extractzvalues`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

M 値で頂点をフィルタ

M 値に基づいて頂点をフィルタリングし、指定した最小値以上、指定した最大値以下の M 値を持つ頂点のみによるジオメトリを返します。

最小値を指定しない場合、最大値のみでフィルタリングします。同様に、最大値を指定しない場合には、最小値のみでフィルタリングします。

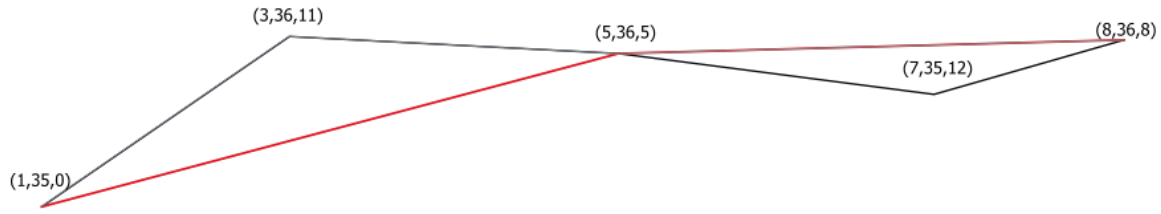


図 27.76: 赤線は、黒線の M 値が 10 以下の頂点のみを表す

M が有効な、ライン、ポリゴン地物の **地物の *In-place* 編集** が可能です

注釈: 入力ジオメトリの属性と使用する最大最小値によっては、このアルゴリズムによって作成される結果のジオメトリは無効なジオメトリとなる場合があります。

参考:

Z **値で頂点をフィルタ** 、 **頂点を抽出** 、 **特定の点を抽出**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	頂点をフィルタリングしたいラインまたはポリゴンの入力ベクタレイヤ
最小値 オプション	MIN	[数値] デフォルト：**未設定**	M 値の下限値
最大値 オプション	MAX	[数値] デフォルト：**未設定**	M 値の上限値
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	フィルタリングされた頂点のみの地物の出力ベクタレイヤ

Python コード

Algorithm ID: native:filterverticesbym

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

Z 値で頂点をフィルタ

Z 値に基づいて頂点をフィルタリングし、指定した最小値以上、指定した最大値以下の Z 値を持つ頂点のみによるジオメトリを返します。

最小値を指定しない場合、最大値のみでフィルタリングします。同様に、最大値を指定しない場合には、最小値のみでフィルタリングします。

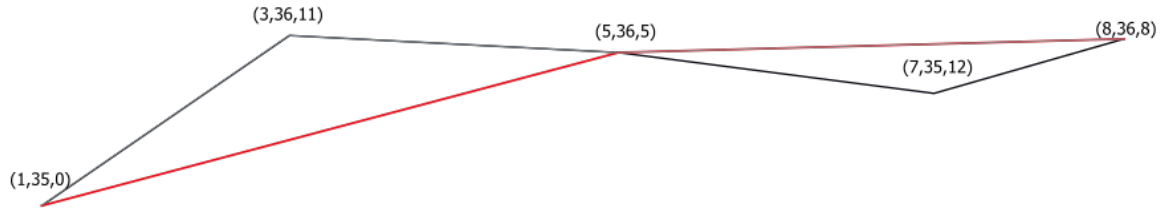


図 27.77: 赤線は、黒線の Z 値が 10 以下の頂点のみを表す

Z が有効な、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

注釈: 入力ジオメトリの属性と使用する最大最小値によっては、このアルゴリズムによって作成される結果のジオメトリは無効なジオメトリとなる場合があります。ジオメトリの有効性を保証するためには、**ジオメトリを修復** アルゴリズムを実行すると良いでしょう。

参考:

M **値で頂点をフィルタ** 、 **頂点を抽出** 、 **特定の点を抽出**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	頂点をフィルタリングしたいラインまたはポリゴンの入力ベクタレイヤ
最小値 オプション	MIN	[数値] デフォルト：**未設定**	Z 値の下限値
最大値 オプション	MAX	[数値] デフォルト：**未設定**	Z 値の上限値
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	フィルタリングされた頂点のみの地物の出力ベクタレイヤ

Python コード

Algorithm ID: native:filterverticesbyz

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリを修復

与えられた無効なジオメトリから、入力された頂点を削除することなく有効な表現を作成しようと試みます。既に有効なジオメトリはそのまま返されます。結果は常にマルチパートジオメトリのレイヤとなります。

M を有効としない、ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

注釈: M 値は出力から削除されます。

参考:

有効性チェック

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
修復方法 NEW in 3.28	METHOD	[列挙型] デフォルト： 1	ジオメトリの修復に使われる方法です。次のいずれかです： • 0 -- 「ラインワーク」: すべてのリングをノードを持ったラインの集合に結合し、そのラインワークから有効なポリゴンを抽出します • 1 --- 「構造」: まず全てのリングを有効なものにし、次にシェルをマージし、シェルからホールを引いて有効な結果を生成します。ホールとシェルが正しく分類されていることを前提とします。GEOS 3.10 以降でビルドされた QGIS のバージョンが必要です（**ヘルプ** *QGIS* **について** メニューを確認してください）。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	修正されたジオメトリの出力ベクタレイヤ

Python コード

Algorithm ID: native:fixgeometries

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

右手ルールを強制

ポリゴンで囲まれた領域が境界の右側にあるという、右手ルール（Right-Hand-Rule）を強制的にジオメトリに適用します。具体的には、外側のリングは時計回りの方向に、内側のリングは反時計回りの方向とします。

ポリゴン地物の **地物の *In-place* 編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	ジオメトリのリングの向きが修正された出力ベクタレイヤ

Python コード

Algorithm ID: `native:forcerhr`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

日付変更線で測地線を分割

ラインが日付変更線（経度 ±180 度線）を横切る場合に、ラインを複数の測地線セグメントに分割します。

経度 180 度線でラインを分割することで、一部の投影法での表示が改善されます。返されるジオメトリは常にマルチパートジオメトリとなります。

入力ジオメトリのラインセグメントが経度 180 度線を跨ぐたびに、セグメントは 2 つに分割されます。切断点の緯度は、このセグメントの両側の点を結ぶ測地線を用いて決定されます。この切断点の計算には、現在のプロジェクトの楕円体設定が使用されます。

入力ジオメトリが M 値や Z 値を持つ場合、経度 180 度線上に作成される新しい頂点では、M 値や Z 値が線形補間されます。

ライン地物の **地物の *In-place* 編集** が可能です。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	経度 180 度線でラインが分割された出力ラインベクタレイヤ

Python コード

Algorithm ID: native:antimeridiansplit

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

式によるジオメトリ

QGIS 式を使用して、入力地物の既存のジオメトリを更新（または新たなジオメトリを作成）します。

これにより、QGIS 式エンジンの柔軟性を存分に活用して出力地物のジオメトリの操作や作成ができ、ジオメトリの複雑な変更が可能になります。

QGIS 式関数のヘルプは、 **式ビルダー** にある組み込みのヘルプを参照してください。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
出力のジオメトリ型	OUTPUT_GEOMETRY	[列挙型] デフォルト： 0	出力ジオメトリは式に強く依存しています。例えば、バッファを作成する場合、ジオメトリタイプはポリゴンでなければなりません。次のいずれかを指定します： • 0 --- ポリゴン（Polygons） • 1 --- ライン • 2 --- 点（Point）
出力ジオメトリは Z 次元を持つ	WITH_Z	[ブール値] デフォルト：False	出力ジオメトリが Z 次元を持つかどうかを選択します
出力ジオメトリは M 値を持つ	WITH_M	[ブール値] デフォルト：False	出力ジオメトリが M 値を持つかどうかを選択します
ジオメトリの式	EXPRESSION	[式] デフォルト：'$geometry'	使用したいジオメトリ式を入力します。ボタンを押すと式ダイアログが開きます。ダイアログには関連する全ての式・関数がリストされ、ヘルプや使用例も確認できます。
出力レイヤ	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：任意]	出力ベクタレイヤ

Python コード

Algorithm ID: `native:geometrybyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線上の等間隔点

ラインまたはカーブラインジオメトリに沿って指定の距離で内挿された位置にポイントジオメトリを作成します。

Z 値や M 値は既存の値から線形補間されます。

マルチパートジオメトリが入力された場合、最初のパートだけが考慮されます。

指定された距離が入力地物の長さよりも大きい場合には、結果となる地物のジオメトリは null です。

図 27.78: ラインの始点から 500m の位置に補間された点

参考:

ジオメトリに沿った点群

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
距離	DISTANCE	[数値] デフォルト： 0.0	ラインの始点からの距離
内挿点	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
内挿点	OUTPUT	[ベクタ：ポイント]	ラインまたはポリゴン境界に沿った指定の距離の位置の出力ポイントベクタレイヤ

Python コード

Algorithm ID: `native:interpolatepoint`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

マルチパートの上位 N 個

ポリゴンまたはマルチポリゴンのレイヤを入力として、マルチポリゴンの各地物について面積の大きな *n* 個のパートのみを残した新しいレイヤを返します。地物のパートが *n* 個以下の場合には、その地物はコピーされるだけです。

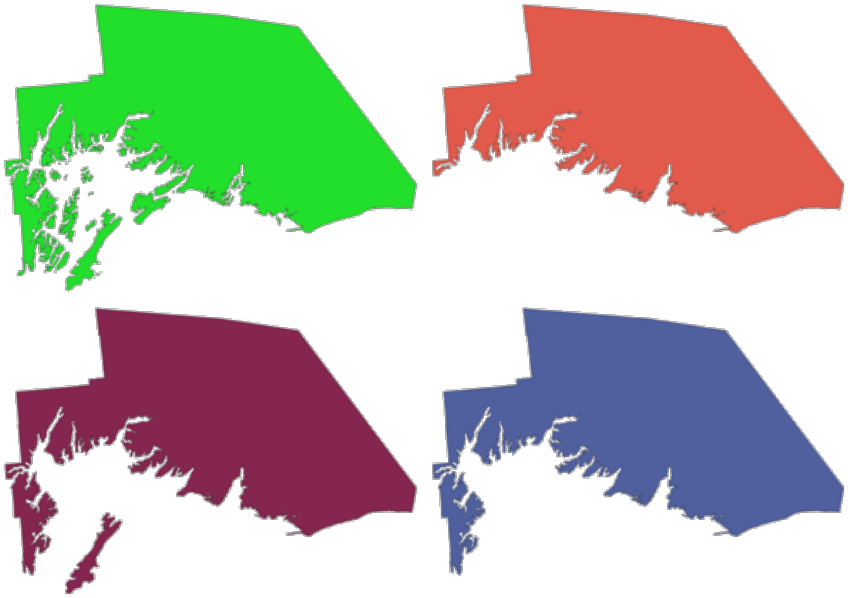


図 27.79: 左上から順に、元のマルチパート地物、面積上位 1 個、 2 個、 3 個のパートのみを残したもの

パラメータ

ラベル	名前	データ型	説明
ポリゴン（Polygons）	INPUT	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
マルチパートの構成部分のうち、面積で上位何個を抜き出すか	PARTS	[数値] デフォルト： 1	残したいパートの数。1 とすると、地物の最大パートのみとなります。
出力ファイル	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力ファイル	OUTPUT	[ベクタ：ポリゴン]	各地物について、面積の大きな N 個のパートの出力ポリゴンベクタレイヤ

Python コード

Algorithm ID: `qgis:keepnbiggestparts`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラインの一部の切り出し

（ラインの始点から測った）指定された開始距離と終了距離の間にあるライン（またはカーブ）の部分を返します。

Z 値や M 値は既存の値から線形補間されます。

マルチパートジオメトリが入力された場合、最初のパートだけが考慮されます。

図 27.80: 開始距離を 0m、終了距離を 250m に設定して切り出されたラインの一部分

ライン地物の **地物の** *In-place* **編集** が可能です。

参考:

線の延長

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
始点の延長分	START_DISTANCE	[数値]	出力地物の始点となる、入力ラインに沿った距離
終点の延長分	END_DISTANCE	[数値]	出力地物の終点となる、入力ラインに沿った距離
線分切り出し (Substring)	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
線分切り出し (Substring)	OUTPUT	[ベクタ：ライン]	出力結果のラインベクタレイヤ

Python コード

Algorithm ID: native:linesubstring

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線をポリゴンに変換（lines to polygons）

入力ラインレイヤのラインをポリゴンのリングとして使用したポリゴンレイヤ作成します。

出力レイヤの属性テーブルは、入力レイヤのものと同じです。

デフォルトメニュー ： **ベクタ　ジオメトリツール**

参考:

ポリゴンを線に変換 、 **ポリゴン化（*Polygonize*）** 、 **ジオメトリタイプの変換**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
ポリゴン（Polygons）	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ポリゴン（Polygons）	OUTPUT	[ベクタ：ポリゴン]	出力結果のポリゴンベクタレイヤ

Python コード

Algorithm ID: `qgis:linestopolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線の結合

マルチラインストリングジオメトリの接続した部分を全て結合させ、単一のラインストリングジオメトリにします。

入力のマルチラインストリングジオメトリの任意の部分が不連続な場合、結果のジオメトリは、結合可能な部分は結合されたラインと、不連続なラインのパートからなるマルチラインストリングになります。

ライン地物の **地物の** *In-place* **編集** が可能です。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	（線が結合された）出力ラインベクタレイヤ

Python コード

Algorithm ID: `native:mergelines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最小境界ジオメトリ

入力レイヤの地物を囲むジオメトリを作成します。地物はフィールドでグループ化することができます。グループ化した場合、出力レイヤにはグループ値ごとに 1 つの地物が含まれます。この地物は、グループ値と一致する値を持つ地物のジオメトリをカバーするジオメトリ（最小バウンディングボックス）を持ちます。

以下の包絡ジオメトリタイプをサポートしています：

- バウンディングボックス（エンベロープ）
- 回転長方形
- 円
- 凸包（convex hull）

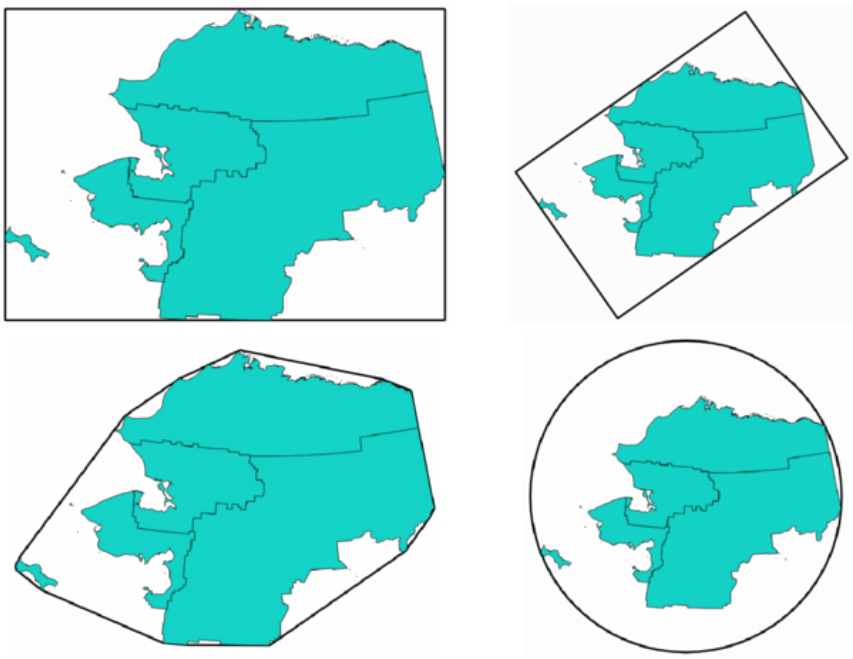


図 27.81: 左上から時計回りに、エンベロープ、回転長方形、円、凸包

参考:

最小包含円

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ

次のページに続く

表 27.149 – 前のページからの続き

ラベル	名前	データ型	説明
属性 オプション	FIELD	[テーブルのフィールド：任意]	地物はあるフィールドでグループ化できます。グループ化した場合、出力レイヤはグループ値ごとに 1 つの地物を含みます。この地物は、グループ値とマッチする地物のみをカバーする最小ジオメトリを持ちます。
ジオメトリタイプ	TYPE	[列挙型] デフォルト：0	包絡ジオメトリのタイプ。次のいずれかです： • 0 --- バウンディングボックス（エンベロープ） • 1 --- 最小の回転長方形 • 2 --- 最小の外接円 • 3 --- 凸包（convex hull）
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	（最小境界の）出力ポリゴンベクタレイヤ

Python コード

Algorithm ID: `qgis:minimumboundinggeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最小包含円

入力レイヤの各地物を覆う最小面積の包含円を計算します。

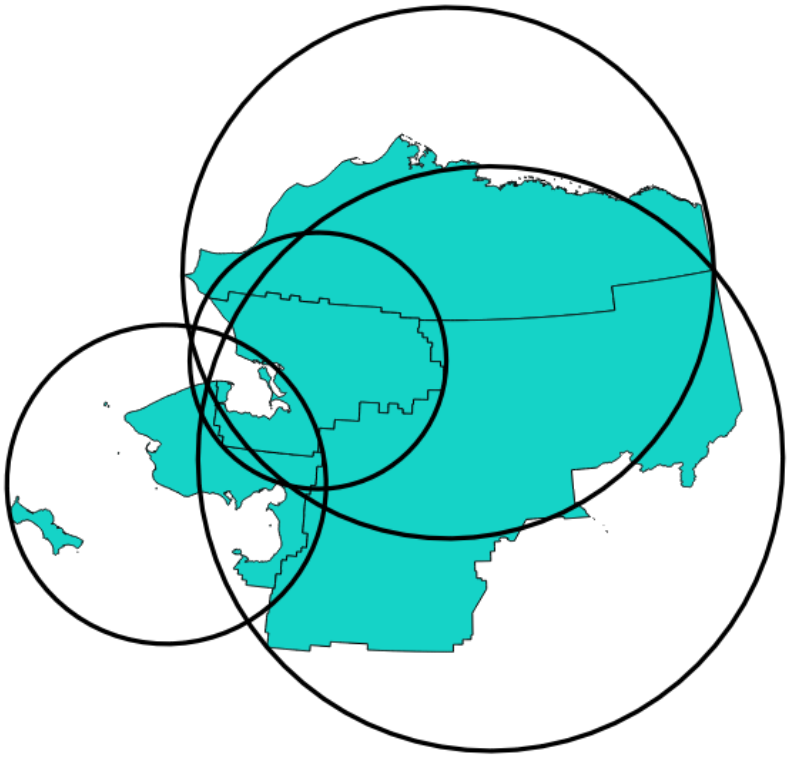


図 27.82: 各地物の包含円

ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

最小境界ジオメトリ

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
円のセグメント数	SEGMENTS	[数値] デフォルト： 72	円を近似するのに使用するセグメントの数。最小値は 8、最大値は 100000。
最小包含円	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
最小包含円	OUTPUT	[ベクタ：ポリゴン]	出力結果のポリゴンベクタレイヤ

Python コード

Algorithm ID: native:minimumenclosingcircle

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

多重リングバッファ

入力レイヤの地物に対して、固定距離や動的距離を使用した、指定する層数分の多重リング（ **ドーナツ** ）バッファを計算します。

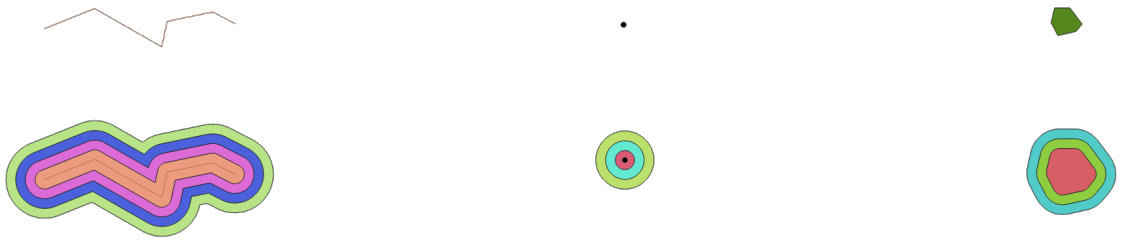


図 27.83: ライン、ポイント、ポリゴンレイヤに対する多重リングバッファ

ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

バッファ（*buffer*） 、 **可変距離バッファ** 、 **長方形・楕円・ダイヤモンド** 、 **片側バッファ**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
リングバッファの層数	RINGS	[数値] デフォルト： 1	リングの数。単一の値（すべての地物に対して同数のリング）とすることもできますし、地物データから取得して設定する（リングの数は地物の値に依存する）こともできます。
リング間の距離（リングの太さ）	DISTANCE	[数値] デフォルト： 1.0	リング間の距離。単一の値（すべての地物に対して同じリング間距離）とすることもできますし、地物データから取得して設定する（距離は地物の値に依存する）こともできます。

次のページに続く

表 27.150 – 前のページからの続き

ラベル	名前	データ型	説明
多重リングバッファ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
多重リングバッファ	OUTPUT	[ベクタ：ポリゴン]	出力結果のポリゴンベクタレイヤ

Python コード

Algorithm ID: `native:multiringconstantbuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

マルチパートをシングルパートに変換

入力レイヤのマルチパート地物をシングルパートの地物に分割します。

出力レイヤの属性は元の属性と同じですが、シングルパート地物にも分割されます。

図 27.84: 左：マルチパートのソースレイヤ、右：シングルパートの出力結果

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： **ベクタ　ジオメトリツール**

参考:

シングルパートをマルチパートに集約 、 **マルチパートに強制変換**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力ベクタレイヤ

Python コード

Algorithm ID: `native:multiparttosingleparts`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

オフセット線

ラインを指定した距離だけオフセットします。正の値の距離はラインを左に、負の値の距離は右にオフセットします。

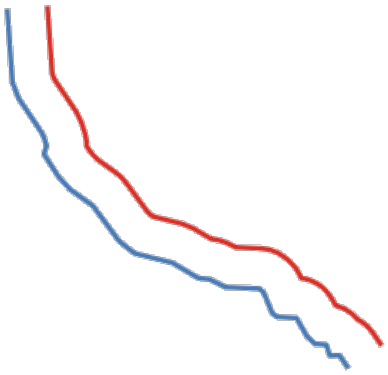


図 27.85: 青色ラインはソースレイヤ、赤色ラインはオフセットしたもの

ライン地物の **地物の** *In-place* **編集** が可能です。

参考:

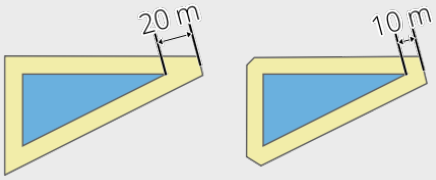
オフセット線の配列 、 **平行移動**（*translate*）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
距離	DISTANCE	[数値] デフォルト： 10.0	オフセット距離。右にある「データによって定義された上書き」ボタンを使用して、距離を計算するためのフィールドを選択できます。これによって、地物によって異なるオフセット距離を使用することができます（ **可変距離バッファ** 参照）。
セグメント	SEGMENTS	[数値] デフォルト： 8	丸みを帯びたオフセットの四分円を近似するために使用するセグメントの数を制御します

次のページに続く

表 27.151 – 前のページからの続き

ラベル	名前	データ型	説明
継ぎ目スタイル	JOIN_STYLE	[列挙型] デフォルト： 0	ラインの角をオフセットする場合に使用する継ぎ目スタイルを round、miter、bevel から指定します。オプションは次のとおり： • 0 --- Round • 1 --- Miter • 2 --- Bevel  図 27.86: round、miter、bevel のつなぎ目スタイル
miter 制限	MITER_LIMIT	[数値] デフォルト： 2.0	miter 継ぎ目を作成する際に使用するオフセットジオメトリからの最大距離を、オフセット距離の係数として設定します（miter 継ぎ目スタイルにのみ適用されます）。最小値: 1.0  図 27.87: 10m バッファで制限 2 の場合と、10m バッファで制限 1 の場合
オフセット	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	（オフセットの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
オフセット	OUTPUT	[ベクタ：ライン]	（オフセットの）出力ラインレイヤ

Python コード

Algorithm ID: `native:offsetline`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最小の回転長方形

入力レイヤの各地物を覆う最小面積の回転長方形を計算します。

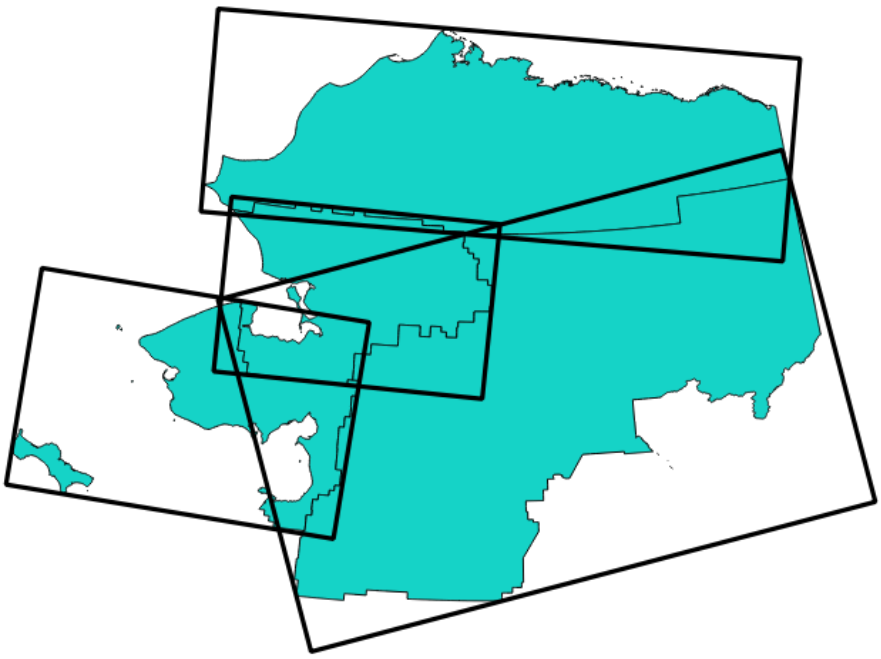


図 27.88: 最小の回転長方形

ポリゴン地物の **地物の *In-place* 編集** が可能です

参考:

最小境界ジオメトリ

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
BBox の出力	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
BBox の出力	OUTPUT	[ベクタ：ポリゴン]	出力結果のポリゴンベクタレイヤ

Python コード

Algorithm ID: native:orientedminimumboundingbox

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリの直交化

入力のラインまたはポリゴンレイヤのジオメトリの直交化を試みます。この処理は、ジオメトリ内のすべての角が直角または直線となるようにジオメトリ内の頂点を移動させます。

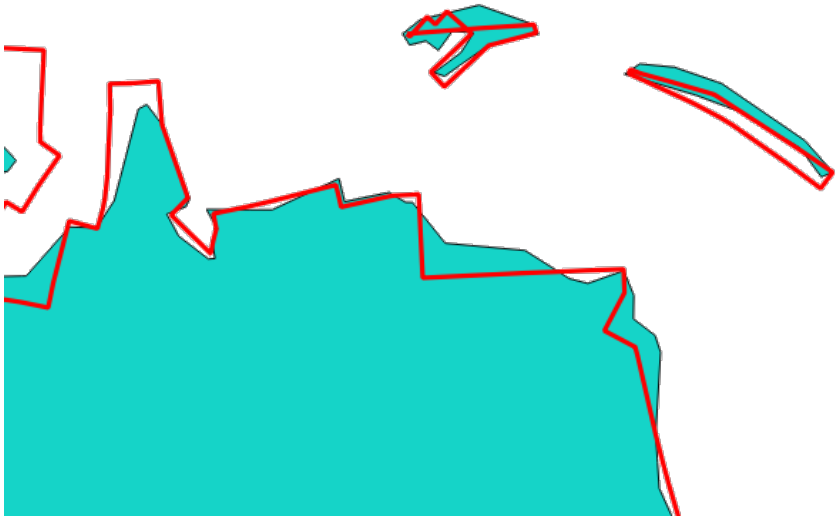


図 27.89: 青色はソースレイヤ、赤色ラインは直交化の結果

☑ ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
最大許容角（度）	ANGLE_TOLERANCE	[数値] デフォルト： 15	頂点が調整されうる直角または直線からの最大偏差を指定します。許容範囲が小さいと、すでに直角に近い頂点だけが調整されます。許容範囲が大きいと、直角から大きく外れた頂点も調整されます。
アルゴリズムの最大反復回数	MAX_ITERATIONS	[数値] デフォルト： 1000	最大反復回数により大きな数を設定すると、処理時間はかかりますが、より直交したジオメトリが得られます。

次のページに続く

表 27.152 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	直交化結果の出力ポリゴンベクタレイヤ

Python コード

Algorithm ID: `native:orthogonalize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

内部保証点（point on surface）

入力レイヤの各地物について、地物ジオメトリ上にあることが保証されたポイントを返します。

ライン地物の **地物の** *In-place* **編集** が可能です

参考:

重心

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
マルチパートの場合に各パートに点を作成	ANGLE_TOLERANCE	[ブール値]	チェックした場合、ジオメトリの各パートにポイントを作成します。
点（Point）	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ポイントベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
点（Point）	OUTPUT	[ベクタ：ポイント]	出力結果のポイントベクタレイヤ

Python コード

Algorithm ID: native:pointonsurface

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリに沿った点群

ラインまたはポリゴンのジオメトリに沿って等間隔のポイント群を作成します。作成したポイントには、ジオメトリに沿った距離やポイント位置でのラインの方向角の属性が新たに追加されます。

オプションで開始と終了のオフセットを指定できます。これは、ポイントを作成するジオメトリの開始位置と終了位置の距離を制御します。

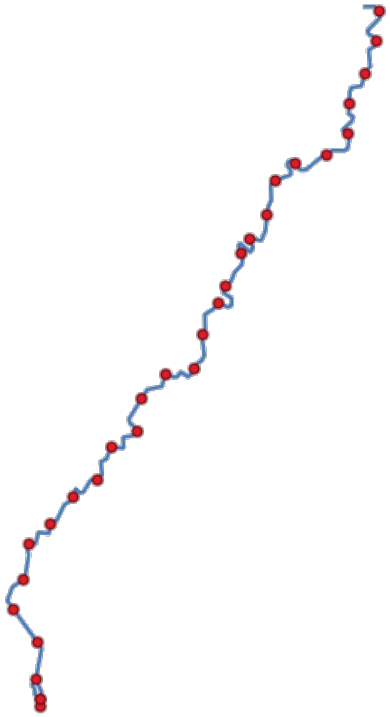


図 27.90: ソースラインレイヤにそって作成されたポイント群

参考:

線上の等間隔点

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
距離	DISTANCE	[数値] デフォルト： 1.0	ラインに沿った 2 つの連続するポイント間の距離
開始オフセット	START_OFFSET	[数値] デフォルト： 0.0	入力ラインの始点からの距離で、最初に作成されるポイントの位置を表します。

次のページに続く

表 27.153 – 前のページからの続き

ラベル	名前	データ型	説明
終了オフセット	END_OFFSET	[数値] デフォルト： 0.0	入力ラインの終点からの距離で、これより先にはポイント地物が作成されません。
内挿点	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
内挿点	OUTPUT	[ベクタ：ポイント]	入力レイヤのラインまたはポリゴン境界に沿って地物が配置されたポイントベクタレイヤ

Python コード

Algorithm ID: native:pointsalonglines

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

近隣点を散らす

指定された近接距離に関して近接するポイント地物を探し出して、それらの重心を中心とする円周上に放射状にポイントを分布させます。重なり合った地物を散らすのに便利なツールです。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
近隣点とみなす距離	PROXIMITY	[数値] デフォルト： 1.0	この距離以下のポイント地物どうしは近接しているとみなします。近接した地物はまとめて散らされます。
近隣点を配置する円の半径	DISTANCE	[数値] デフォルト： 1.0	近接した地物が配置される円の半径
2 点の場合は水平方向に散らす	HORIZONTAL	[ブール値] デフォルト： False	近接したポイントが 2 点だけの場合には、円周上で垂直方向ではなく水平方向に配置します。
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	出力ポイントベクタレイヤ

Python コード

Algorithm ID: `qgis:pointsdisplacement`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

到達不能極（PIA）

ポリゴンレイヤについて、到達不能極、すなわちポリゴンの境界から最も離れた内部点を計算します。

このアルゴリズムでは、 'polylabel' アルゴリズム（Vladimir Agafonkin, 2016）を使用します。これは、指定した許容範囲内で真の到達不能極を見つけることが保証された反復的なアプローチのアルゴリズムです。より正確な許容範囲（低い値）では、より多くの反復回数を必要とし、計算時間が長くなります。

計算された極からポリゴン境界までの距離は、出力レイヤで新しい属性として保存されます。

図 27.91: 到達不能極（PIA）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ベクタレイヤ
許容範囲	TOLERANCE	[数値] デフォルト： 1.0	計算上の許容範囲の設定
点（Point）	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
点（Point）	OUTPUT	[ベクタ：ポイント]	出力結果のポイントベクタレイヤ

Python コード

Algorithm ID: `native:poleofinaccessibility`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ポリゴン化（Polygonize）

ポリゴンレイヤを作成します。レイヤの地物の境界は、ラインレイヤの **閉じた** 地物から生成されます。

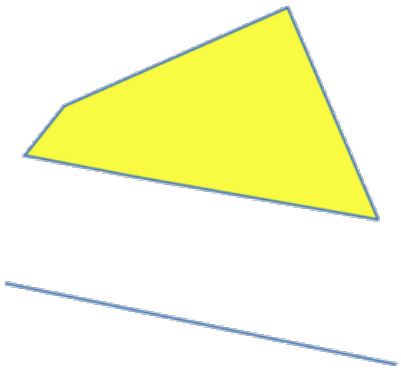


図 27.92: 閉じたラインから生成されたポリゴン（黄色）

注釈: ポリゴンに変換するため、ラインレイヤは閉じた形でなければなりません。

参考:

ポリゴンを線に変換 、 **線をポリゴンに変換**（*lines to polygons*） 、 **ジオメトリタイプの変換**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
ラインレイヤのテーブル構造を維持する オプション	KEEP_FIELDS	[ブール値] デフォルト：False	チェックを入れると、入力レイヤのフィールドを引き継ぎます（テーブル構造のみで、値は引き継ぎません）
ポリゴン（Polygons）	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ポリゴンベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ポリゴン（Polygons）	OUTPUT	[ベクタ：ポリゴン]	ラインから生成された出力結果のポリゴンベクタレイヤ

Python コード

Algorithm ID: `native:polygonize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ポリゴンを線に変換

ポリゴンレイヤを入力として、ラインレイヤを作成します。ラインは入力レイヤ内のポリゴンの境界を表します。

出力レイヤの属性テーブルは、入力レイヤのものと同じです。

図 27.93: 黒色のラインはアルゴリズムの結果

デフォルトメニュー ： ベクタ　ジオメトリツール

参考:

線をポリゴンに変換（*lines to polygons*） 、 **ポリゴン化（*Polygonize*）** 、 **ジオメトリタイプの変換**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
線レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
線レイヤ	OUTPUT	[ベクタ：ライン]	ポリゴンから生成された出力結果のラインベクタレイヤ

Python コード

Algorithm ID: native:polygonstolines

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

投影点（デカルト座標）

指定した距離と方向（方位角）でポイントジオメトリを投影します。

ライン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
方位（北を 0 とする角度）	BEARING	[数値] デフォルト： 0.0	北向きから時計回りに測った角度。単位は度 (°)
距離	DISTANCE	[数値] デフォルト： 1.0	ジオメトリをオフセットさせる距離。長さはレイヤの単位。
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力ポイントベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	（投影された）出力ポイントベクタレイヤ

Python コード

Algorithm ID: `native:projectpointcartesian`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

マルチパートに強制変換

シングルパートジオメトリのベクタレイヤを入力として、すべてのジオメトリがマルチパートのベクタレイヤを新たに作成します。

すでにマルチパートの入力地物はそのままで変更はありません。

このアルゴリズムを使用すると、マルチパート地物を必要とするデータプロバイダに対応するために、ジオメトリを強制的にマルチパートタイプに変換することができます。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

集計 、 **シングルパートをマルチパートに集約**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力結果のマルチパートベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力結果のマルチパートベクタレイヤ

Python コード

Algorithm ID: `native:promotetomulti`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

長方形・楕円・ダイヤモンド

入力ポイントレイヤの各地物について、長方形、楕円、またはダイヤモンド形のバッファ領域を作成します。

形状パラメータは全ての地物について一定値とすることもできますが、フィールドや式を使用して動的に設定することもできます。

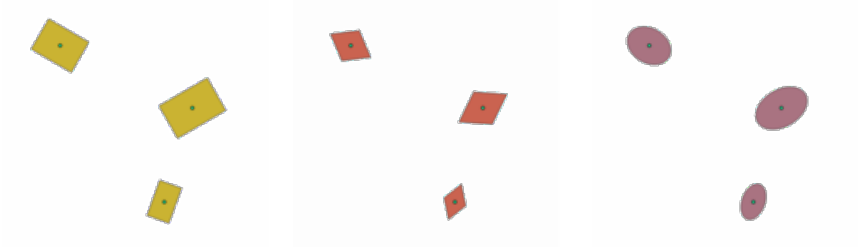


図 27.94: 動的パラメータを使用した、さまざまなバッファ形状

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
Shape	SHAPE	[列挙型]	使用する形状。次のいずれかです： • 0 --- 四角形 • 1 --- 楕円形（Oval） • 2 --- ダイヤモンド（Diamond）
幅	WIDTH	[数値] デフォルト： 1.0	バッファ形状の幅
高さ	HEIGHT	[数値] デフォルト： 1.0	バッファ形状の高さ

次のページに続く

表 27.156 – 前のページからの続き

ラベル	名前	データ型	説明
回転 オプション	ROTATION	[数値] デフォルト： なし	バッファ形状の回転量
セグメント	SEGMENTS	[数値] デフォルト： 36	円全体のセグメント数（ **楕円** 形状で使用）
ポリゴン（Polygon）	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ポリゴン（Polygon）	OUTPUT	[ベクタ：ポリゴン]	（バッファ形状の）出力ベクタレイヤ

Python コード

Algorithm ID: native:rectanglesovalsdiamonds

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

重複する頂点の削除

頂点を削除してもジオメトリが縮退しないような重複した頂点を地物から削除します。

許容範囲パラメータは、頂点が重複しているかを判定する際の座標の許容誤差を指定します。

デフォルトでは、重複を検出する際にZ値は考慮しません。例えば、同じXY座標でZ値は異なる2つの頂点は、やはり重複しているとみなされ、頂点の1つは削除されます。 **Z値を使用する** パラメータがtrueの場合にはZ値も比較され、XとYは同じだがZは異なる頂点は削除されなくなります。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

注釈: 頂点の重複は、マルチパートジオメトリの異なるパート間ではチェックされません。例えば、ポイントが重複しているマルチポイントジオメトリは、このアルゴリズムでは変更されません。

参考:

頂点を抽出 、 **特定の点を抽出** 、 **重複ジオメトリの削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
許容範囲	TOLERANCE	[数値] デフォルト：0.000001	指定した距離よりも近い頂点は重複しているとみなします
Z値を使用する	USE_Z_VALUE	[ブール値] デフォルト：False	**Z値を使用する** パラメータがtrueの場合にはZ値も比較され、XとYは同じだがZは異なる頂点は削除されなくなります。
クリーニング済み出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackageに保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
クリーニング済み出力	OUTPUT	[入力レイヤと同じ]	（重複する頂点が削除された）出力ベクタレイヤ

Python コード

Algorithm ID: native:removeduplicatevertices

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

NULL ジオメトリの削除

ジオメトリを持たない地物をベクタレイヤから削除します。ジオメトリを持っている地物はそのままコピーされます。

NULL ジオメトリの地物は別のレイヤに保存できます。

空ジオメトリも削除する にチェックを入れると、このアルゴリズムは座標を持っていない、すなわちジオメトリが空の地物を削除します。この場合、「NULL ジオメトリの出力レイヤ」にもこのオプションが反映され、レイヤには NULL ジオメトリの地物と空のジオメトリの地物の両方が含まれるようになります。

参考:

重複ジオメトリの削除

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	（NULL でないジオメトリを含む）入力ベクタレイヤ
空ジオメトリも削除する	REMOVE_EMPTY	[ブール値]	

次のページに続く

表 27.158 – 前のページからの続き

ラベル	名前	データ型	説明
NULL ジオメトリでない地物の出力レイヤ	OUTPUT オプション	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	NULL でない（かつ空でない）ジオメトリの出力ベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
NULL ジオメトリの出力レイヤ オプション	NULL_OUTPUT	[入力レイヤと同じ] デフォルト：[出力をスキップ]	NULL ジオメトリ（あるいは空ジオメトリ）の出力ベクタレイヤを指定します。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
NULL ジオメトリの出力レイヤ	NULL_OUTPUT	[入力レイヤと同じ]	（NULL ジオメトリの、オプションを選択した場合には空ジオメトリも含む）出力ベクタレイヤ
NULL ジオメトリでない地物の出力レイヤ	OUTPUT	[入力レイヤと同じ]	（NULL ジオメトリではない、かつオプションを選択した場合には空ジオメトリでもない）出力ベクタレイヤ

Python コード

Algorithm ID: `native:removenullgeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線の向きの反転

ラインレイヤの向きを反転します。

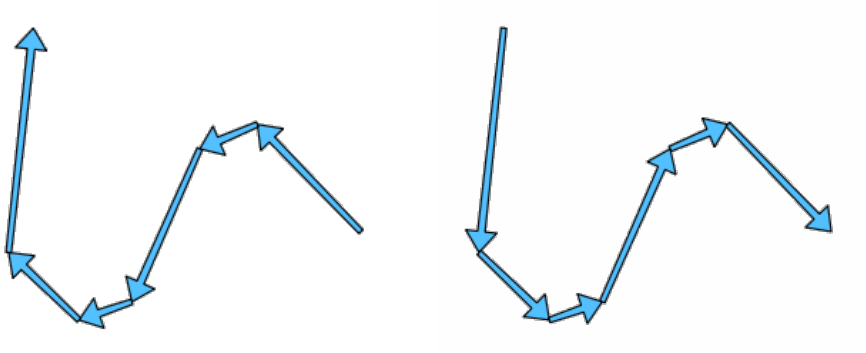


図 27.95: 線の向きの反転前と反転後

ライン地物の **地物の** *In-place* **編集** が可能です。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	（線の向きが反転した）出力ラインベクタレイヤ

Python コード

Algorithm ID: native:reverselinedirection

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリの回転

地物ジオメトリを指定された角度だけ時計回りに回転させます。回転は各地物の重心を中心に行われますが、オプションとして指定した単一の点を中心に回転させることもできます。

ポイント、ライン、ポリゴン地物の **地物の *In-place* 編集** が可能です

参考:

平行移動（*translate*） 、 **座標の *XY* 入れ替え**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
回転する角度（時計回り、単位は度）	ANGLE	[数値] デフォルト： 0.0	度単位で表した回転角度
回転の中心 オプション	ANCHOR	[ポイント] デフォルト： なし	地物の回転中心となる点の XY 座標。指定されていない場合は、各地物の重心の周りに回転します。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（ジオメトリが回転した）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	回転したジオメトリの出力ベクタレイヤ

Python コード

Algorithm ID: native:rotatefeatures

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

丸み係数

NEW in 3.24

各地物の丸み係数を計算して新しいフィールドに格納します。入力ベクタレイヤはポリゴンを格納している必要があります。

ポリゴンの丸み係数は、4 π × ポリゴンの面積 / 周囲長 ^{2} と定義されます。丸み係数の値は 0 と 1 の間です。完全な円の丸み係数は 1 で、完全に平らなポリゴンの丸み係数は 0 です。

注釈: アルゴリズムはマルチパートポリゴン地物に NULL を返します。

ポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ベクタレイヤ
丸み係数	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（丸み係数フィールドを持った）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	丸み係数値をフィールドに持った、出力ベクタレイヤ

Python コード

Algorithm ID: `native:roundness`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

曲線のセグメント化（角度）

曲線部分を直線化することで、ジオメトリをセグメント化します。

セグメント化は、直線化されるジオメトリ上の頂点間の最大許容半径角度（元の円弧の曲率中心と、直線化されたジオメトリ上の連続する出力頂点がなす角度）を指定することによって実行されます。湾曲していないジオメトリはそのまま、変更されません。

参考:

曲線のセグメント化（距離） 、 **ジオメトリの簡素化** 、 **スムージング**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
セグメントの最大許容角度（単位は度）	ANGLE	[数値] デフォルト： 5.0	直線化されたジオメトリ上の頂点間の最大許容中心角
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（セグメント化されたジオメトリの）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	セグメント化されたジオメトリの出力ベクタレイヤ

Python コード

Algorithm ID: `native:segmentizebymaxangle`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

曲線のセグメント化（距離）

曲線部分を直線化することで、ジオメトリをセグメント化します。

セグメント化は、元の曲線とセグメント化した直線との最大許容距離を指定することによって実行されます。湾曲していないジオメトリはそのまま、変更されません。

参考:

曲線のセグメント化（角度） 、 **ジオメトリの簡素化** 、 **スムージング**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
最大許容距離	DISTANCE	[数値] デフォルト： 1.0	元の曲線とセグメント化された直線の間の最大許容オフセット距離（レイヤの距離単位）
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（セグメント化されたジオメトリの）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	セグメント化されたジオメトリの出力ベクタレイヤ

Python コード

Algorithm ID: `native:segmentizebymaxdistance`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

M 値の設定

レイヤのジオメトリに M 値を設定します。

レイヤに M 値が既にある場合には、新しい値で上書きされます。レイヤが M 値を持たない場合には、ジオメトリが M 値を持つように更新され、すべてのジオメトリの M 値を指定した値で初期化します。

M が有効な、ポイント、ライン、ポリゴン地物の **地物の *In-place* 編集** が可能です

Tip: 地物情報表示 ボタンを使用すると、追加された M 値を確認できます。値は **地物情報** ダイアログ内にあります。

参考:

ドレープ（ラスタ値を *M* 値に代入） 、 **Z 値の設定** 、 **M 値 Z 値の削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
M 値	M_VALUE	[数値] デフォルト： 0.0	地物ジオメトリに設定する M 値
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（M 値がジオメトリに設定された）出力ベクタレイヤ

Python コード

Algorithm ID: `native:setmvalue`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ドレープ（ラスタ値を M 値に代入）

ラスタレイヤ内のバンドからサンプリングした値を使用して、地物ジオメトリが重なる頂点すべてに M 値を設定します。ラスタ値は、オプションでプリセット量によりスケーリングできます。

レイヤに M 値が既にある場合には、新しい値で上書きされます。レイヤが M 値を持たない場合には、ジオメトリが M 値を持つように更新されます。

M が有効な、ポイント、ライン、ポリゴン地物の **地物の *In-place* 編集** が可能です

参考:

ドレープ（ラスタ値を *Z* 値に代入） 、 **M 値の設定**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
ラスタレイヤ	RASTER	[ラスタ]	M 値となるラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	M 値を取得するラスタのバンド
ラスタの nodata 部分もしくは外側にある頂点の値	NODATA	[数値] デフォルト： 0.0	ラスタ（の有効なピクセル）と交差しない頂点に使用する値
縮尺係数	SCALE	[数値] デフォルト： 1.0	縮尺係数：バンド値にこの値が掛かります。
オフセット NEW in 3.28	OFFSET	[数値] デフォルト： 0.0	オフセット値：「縮尺係数」を適用した後、代数的にバンド値に加算されます。

次のページに続く

表 27.159 – 前のページからの続き

ラベル	名前	データ型	説明
ドレープ出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（ラスタレイヤから M 値を設定した）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ドレープ出力	OUTPUT	[入力レイヤと同じ]	（M 値を設定した）出力ベクタレイヤ

Python コード

Algorithm ID: `native:setmfromraster`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

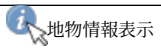
algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

Z 値の設定

レイヤのジオメトリに Z 値を設定します。

レイヤに Z 値が既にある場合には、新しい値で上書きされます。レイヤが Z 値を持たない場合には、ジオメトリが Z 値を持つように更新され、すべてのジオメトリの Z 値を指定した値で初期化します。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

Tip: 

ボタンを使用すると、追加された Z 値を確認できます。値は **地物情報** ダイアログ内にあります。

参考:

ドレープ（ラスタ値を *Z* 値に代入） 、 **M 値の設定** 、 **M 値 Z 値の削除**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
Z 値	Z_VALUE	[数値] デフォルト： 0.0	地物ジオメトリに設定する Z 値
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（Z 値を設定した）出力ベクタレイヤ

Python コード

Algorithm ID: `native:setzvalue`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリの簡素化

ラインまたはポリゴンレイヤのジオメトリを簡素化します。このアルゴリズムは入力レイヤと同じ地物の新しいレイヤを作成しますが、ジオメトリの頂点数は入力レイヤよりも少なくなります。

このアルゴリズムでは、距離ベース（"Douglas-Peucker" アルゴリズム）、面積ベース（"Visvalingam" アルゴリズム）、グリッドにスナップの中から簡素化の方法を選択できます。

図 27.96: 左上から時計回りに、ソースレイヤと、簡素化許容値を順に大きくした結果

ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： ベクタ　 ジオメトリツール

参考:

スムージング 、 **頂点の高密度化（個数ベース）** 、 **間隔による高密度化**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
簡素化の方法	METHOD	[列挙型] デフォルト： 0	簡素化の方法。次のいずれかです： • 0 --- 距離ベース（Douglas-Peucker） • 1 --- グリッドにスナップ • 2 --- 面積ベース（Visvalingam）
許容範囲	TOLERANCE	[数値] デフォルト： 1.0	許容量の閾値（レイヤの距離単位）。2つの頂点間の距離が許容値以下の場合には、セグメントは簡素化され、頂点が削除されます。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（簡素化された）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（簡素化された）出力ベクタレイヤ

Python コード

Algorithm ID: `native:simplifygeometries`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されるIDです。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Pythonコンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

片側バッファ

ラインに対して、ラインの片側のみに指定した距離のバッファを作成します。

バッファは常にポリゴンレイヤとなります。

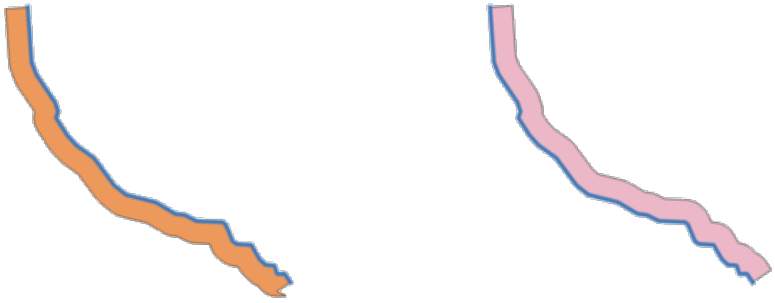


図 27.97: 同じベクタラインレイヤに対する左側バッファと右側バッファ

参考:

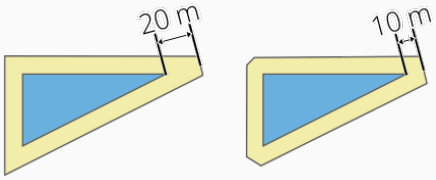
バッファ（*buffer*）

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
距離	DISTANCE	[数値] デフォルト： 10.0	バッファの距離
位置（どちら側か）	SIDE	[列挙型] デフォルト： 0	どちら側にバッファを作成するか。次のいずれかです： • 0 -- 左 • 1 -- 右
セグメント	SEGMENTS	[数値] デフォルト： 8	丸みを帯びたオフセットの四分円を近似するために使用するセグメントの数を制御します

次のページに続く

表 27.161 – 前のページからの続き

ラベル	名前	データ型	説明
継ぎ目スタイル	JOIN_STYLE	[列挙型] デフォルト： 0	ラインの角をオフセットする場合に使用する継ぎ目スタイルを round、miter、bevel から指定します。オプションは次のとおり： • 0 --- Round • 1 --- Miter • 2 --- Bevel  図 27.98: round、miter、bevel のつなぎ目スタイル
miter 制限	MITER_LIMIT	[数値] デフォルト： 2.0	miter 継ぎ目を作成する際に使用するオフセットジオメトリからの最大距離を、オフセット距離の係数として設定します（miter 継ぎ目スタイルにのみ適用されます）。最小値: 1.0  図 27.99: 10m バッファで制限 2 の場合と、10m バッファで制限 1 の場合
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（バッファの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	（バッファの）出力ポリゴンレイヤ

Python コード

Algorithm ID: `native:singlesidedbuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

スムージング

地物ジオメトリに **頂点や角** をさらに追加することによって、ラインまたはポリゴンレイヤのジオメトリをスムージングします。

反復パラメータは、各ジオメトリに適用されるスムージングの反復回数を指定します。反復回数が多いほどジオメトリは滑らかになりますが、ジオメトリ内のノード数は多くなります。

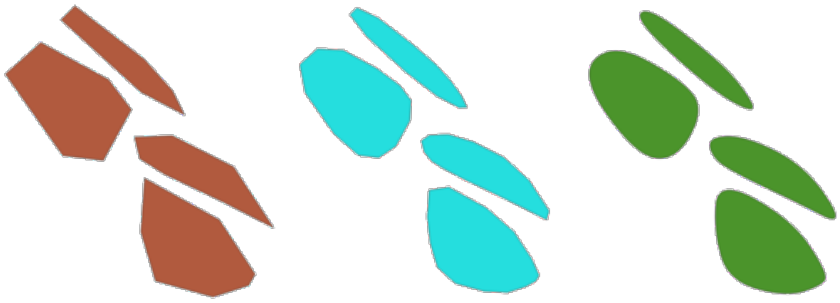


図 27.100: 反復回数を増加させるとジオメトリはより滑らかになる

オフセットパラメータは、平滑化されたジオメトリが元のジオメトリにどれだけ「ぴったりと」追従するかを制御します。値を小さくするとよりぴったりと合い、値を大きくすると緩めのフィットとなります。

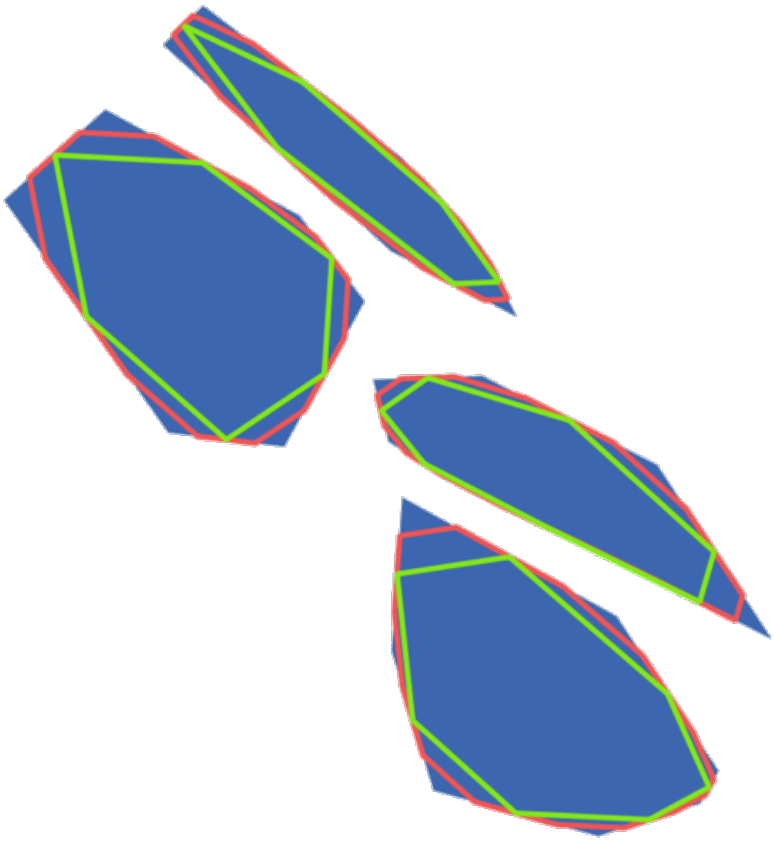


図 27.101: 青：入力レイヤ。赤色線はオフセット値 0.25、緑色線はオフセット値 0.50 の結果

スムージングの最大角度パラメータを使用すると、角度の大きい頂点はスムージングしないようにできます。セグメントの角度がこれより大きいノードは平滑化されません。例えば最大角度を 90 度以下に設定すると、ジオメトリの直角部分は維持されます。

ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

ジオメトリの簡素化 、 **頂点の高密度化（個数ベース）** 、 **間隔による高密度化**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	ラインまたはポリゴンの入力ベクタレイヤ
反復	ITERATIONS	[数値] デフォルト： 1	反復回数を増加させるとジオメトリはより滑らかになります（頂点数が増えます）

次のページに続く

表 27.162 – 前のページからの続き

ラベル	名前	データ型	説明
オフセット	OFFSET	[数値] デフォルト： 0.25	この値を大きくすると、スムージングされたラインまたは境界は入力のラインまたは境界からさらに離れた場所に **移動** します
スムージングの最大角度	MAX_ANGLE	[数値] デフォルト：180.0	この値よりも小さい角度の頂点がスムージングされます
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（スムージングされた）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（スムージングされた）出力ベクタレイヤ

Python コード

Algorithm ID: native:smoothgeometry

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリをレイヤにスナップ

レイヤ内のジオメトリを他のレイヤまたは同じレイヤのジオメトリにスナップさせます。

マッチングは許容距離に基づいて行われ、ジオメトリが参照ジオメトリにマッチするように、必要に応じて頂点が追加または削除されます。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

点をグリッドにスナップ

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
スナップで参照するレイヤ	REFERENCE_LAYER	[ベクタ：任意]	スナップさせたいベクタレイヤ
許容範囲	TOLERANCE	[数値] デフォルト： 10.0	入力レイヤの頂点が参照レイヤのジオメトリにどれだけ近ければスナップさせるかを制御します

次のページに続く

表 27.163 – 前のページからの続き

ラベル	名前	データ型	説明
動作ルール	BEHAVIOR	[列挙型] デフォルト： 0	スナップは既存の頂点またはセグメント（移動させたい頂点に最も近いセグメント上の点）に対して実行されます。利用可能なスナップオプションは以下のとおりです： • 0 --- ノードの整列を優先し、必要に応じて頂点を追加する セグメントが頂点よりも近い場合でも、頂点にスナップすることを優先します。許容範囲内であれば、ジオメトリが正確に追従するように新しい頂点が挿入されます。 • 1 --- 最近傍点を優先し、必要に応じて頂点を追加する 頂点やセグメントに関係なく、最も近い点にスナップします。許容範囲内であれば、ジオメトリが正確に追従するように新しい頂点が挿入されます。 • 2 --- ノードの整列を優先するが、頂点は追加しない セグメントが頂点よりも近い場合でも、頂点にスナップすることを優先します。新しい頂点は追加されません。 • 3 --- 最近傍点を優先するが、頂点は追加しない 頂点やセグメントに関係なく、最も近い点にスナップします。新しい頂点は追加されません。 • 4 --- エンドポイントのみ移動し、ノードの整列を優先する ラインの始点・終点のみスナップ（ポイント地物もスナップします。ポリゴン地物は変更されません）し、頂点にスナップすることを優先します。 • 5 --- エンドポイントのみ移動し、最近傍点を優先する ラインの始点・終点のみスナップ（ポイント地物もスナップします。ポリゴン地物は変更されません）し、最も近い点にスナップします。 • 6 --- エンドポイントはエンドポイントだけにスナップする ラインの始点・終点のみ、他のラインの始点・終点にスナップさせます。 • 7 --- アンカーノードにスナップす

表 27.163 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（スナップした）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（スナップした）出力ベクタレイヤ

Python コード

Algorithm ID: native:snapgeometries

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

点をグリッドにスナップ

ベクタレイヤのジオメトリの座標を変更し、すべてのポイントまたは頂点をグリッドの最近傍点にスナップさせます。

スナップしたジオメトリが計算できない（またはジオメトリが完全に潰れてしまう）場合には、地物のジオメトリが消去されます。

スナップは X 軸、Y 軸、Z 軸、M 値に基づいて実行できます。任意の軸のグリッド間隔を 0 とすると、その軸ではスナップしません。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

注釈: 一部の例外的なケースでは、グリッドへのスナップによって不正なジオメトリが生じる場合があります。

参考:

ジオメトリをレイヤにスナップ

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
X 軸のグリッド間隔	HSPACING	[数値] デフォルト： 1.0	X 軸方向のグリッドの間隔
Y 軸のグリッド間隔	VSPACING	[数値] デフォルト： 1.0	Y 軸方向のグリッドの間隔
Z 軸のグリッド間隔	ZSPACING	[数値] デフォルト： 0.0	Z 軸方向のグリッドの間隔
M 値のグリッド間隔	MSPACING	[数値] デフォルト： 0.0	M 値のグリッドの間隔
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（スナップした）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（スナップした）出力ベクタレイヤ

Python コード

Algorithm ID: `native:snappointstogrid`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

最大長で線を切断

ライン（またはカーブ）を入力として、各地物を複数のパートに分割します。各パートの長さは、指定された最大長さです。分割された新しいラインのパートの始点と終点における Z 値や M 値は、既存の値から線形補間されます。

ライン地物の **地物の** *In-place* **編集** が可能です。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
線の最大長	LENGTH	[数値] デフォルト： 10.0	出力結果のラインの最大長さ
出力レイヤ	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ライン]	新しいラインベクタレイヤ。各地物ジオメトリの長さは、LENGTH パラメータで指定した長さ以下です。

Python コード

Algorithm ID: `native:splitlinesbylength`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリの空間分割

ジオメトリを空間的に分割します。返されるジオメトリは、元のジオメトリを分割したものを含むコレクションとなり、各部分は指定した最大頂点数以下となります。

このアルゴリズムは、ある複雑なジオメトリをより単純な部分に分割する際に役立ちます。これにより空間インデックスが簡単になり、空間演算をより高速に実行できます。カーブジオメトリは、空間分割の前にセグメント化されます。

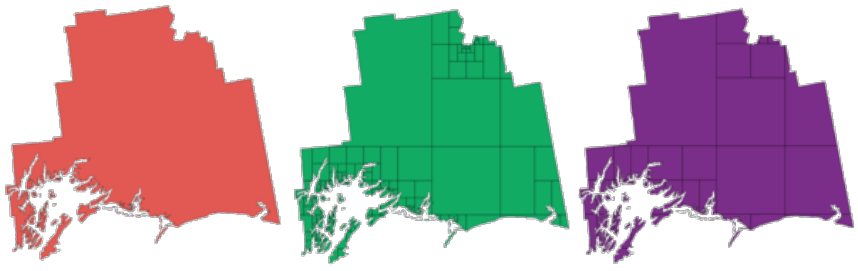


図 27.102: 左：入力レイヤ、中：最大頂点数 100、右：最大頂点数 200

ポイント、ライン、ポリゴン地物の **地物の *In-place* 編集** が可能です

注釈: ジオメトリの空間分割によって、無効なジオメトリパートが生成されたり、自己交差を含む場合があります。

参考:

線をセグメントに分解 、 **ラインの一部の切り出し**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
各部分の最大頂点数	MAX_NODES	[数値] デフォルト： 256	新しい各ジオメトリパートが持つことのできる最大頂点数。値を大きくすると、**ジオメトリパート** の数は少なくなります。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	（空間分割された）出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力ベクタレイヤ

Python コード

Algorithm ID: `native:subdivide`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

座標の XY 入れ替え

入力ジオメトリの X 座標と Y 座標の値を入れ替えます。

このアルゴリズムは、誤って緯度と経度の値を逆にしてしまったジオメトリの修復に使えます。

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

平行移動（*translate***）** 、 **ジオメトリの回転**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	（座標の XY を入れ替えた）出力ベクタレイヤ

Python コード

Algorithm ID: `native:swapxy`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

テイパードバッファ

指定された始点と終点のバッファ直径を使用して、ラインジオメトリに沿って幅が変化するバッファを作成します。

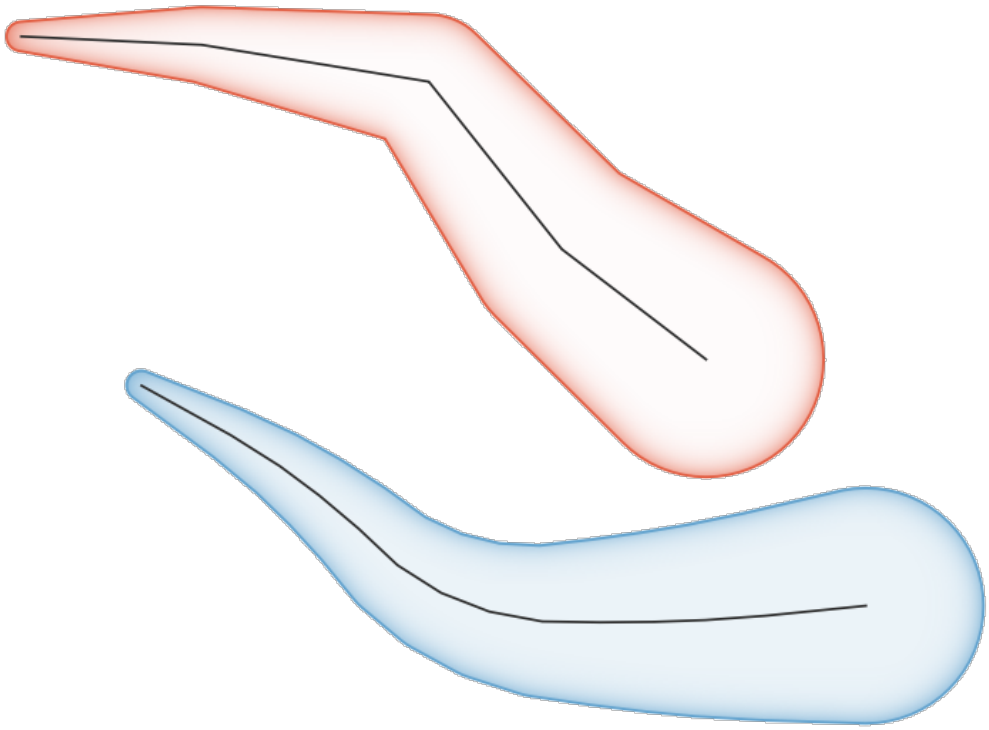


図 27.103: テイパードバッファの例

参考:

可変幅バッファ（*M* 値使用） 、 **バッファ（*buffer*）** 、 **ウェッジバッファを作成**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
始点における直径	START_WIDTH	[数値] デフォルト： 0.0	ライン地物の始点におけるバッファの直径
終点における直径	END_WIDTH	[数値] デフォルト： 0.0	ライン地物の終点におけるバッファの直径
セグメント	SEGMENTS	[数値] デフォルト： 16	丸みを帯びたオフセットの四分円を近似するために使用するセグメントの数を制御します

次のページに続く

表 27.165 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（バッファの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	（バッファの）出力ポリゴンレイヤ

Python コード

Algorithm ID: `native:taperedbuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

テッセレーション

ポリゴンジオメトリのレイヤをテッセレーションし、三角形の構成要素に分割します。

出力レイヤは、各入力地物に関するマルチポリゴンジオメトリのレイヤです。各マルチポリゴンは、複数の三角形ポリゴン要素で構成されます。

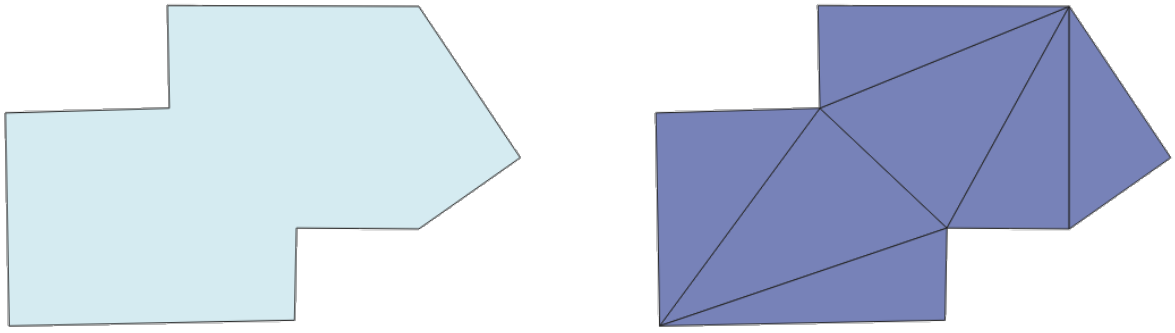


図 27.104: テッセレーションされたポリゴン（右）

ポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポリゴン]	入力ポリゴンベクタレイヤ
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	出力のマルチポリゴン（MultiPolygonZ）レイヤ

Python コード

Algorithm ID: `3d:tessellate`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

トランセクト

（マルチ）ラインストリングの頂点にトランセクトを作成します。

トランセクトとは、入力されたポリライン（の頂点位置）に対してある角度（デフォルトでは垂直）を向いた線のことです。

入力地物のフィールドはトランセクトのフィールドにも引き継がれ、以下の新しいフィールドも追加されます：

- TR_FID: 元の地物の ID
- TR_ID: トランセクトの ID。各トランセクトは一意な ID を持ちます
- TR_SEGMENT: ラインストリングのセグメントの ID
- TR_ANGLE: 頂点位置での元のラインからの角度（度単位）
- TR_LENGTH: トランセクトの全長
- TR_ORIENT: トランセクトを作成する側（ラインの左、右または両側）

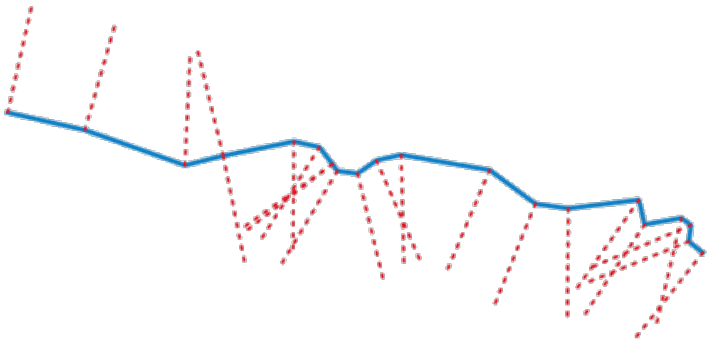


図 27.105: 赤点線は入力ラインレイヤのトランセクト

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
トランセクトの長さ	LENGTH	[数値] デフォルト： 5.0	マップ単位で表したトランセクトの長さ
元の線との角度（度）	ANGLE	[数値] デフォルト： 90.0	トランセクトの角度が変更できます
トランセクトを作成する側（線の進行方向から見て）	SIDE	[列挙型]	トランセクトを作成する側。オプションは次のとおり： • 0 --- 左 • 1 --- 右 • 2 --- 両側
トランセクト	OUTPUT	[ベクタ：ライン] デフォルト：[一時レイヤを作成]	出力ラインレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
トランセクト	OUTPUT	[ベクタ：ライン]	出力のラインレイヤ

Python コード

Algorithm ID: `native:transect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

平行移動（translate）

あらかじめ定義した X、Y の変位分だけオフセットすることで、レイヤ内のジオメトリを移動させます。ジオメトリにある Z 座標や M 値も移動させることができます。

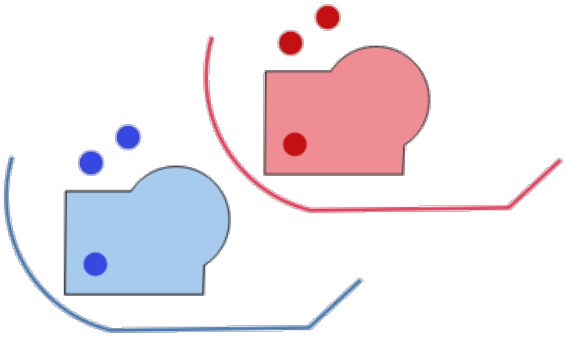


図 27.106: 点線は入力レイヤのジオメトリを平行移動させたものを表す

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

参考:

平行移動した地物の配列 、 **オフセット線** 、 **ジオメトリの回転** 、 **座標の** *XY* **入れ替え**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
オフセット距離（X 軸）	DELTA_X	[数値] デフォルト： 0.0	X 軸方向に適用する移動量
オフセット距離（Y 軸）	DELTA_Y	[数値] デフォルト： 0.0	Y 軸方向に適用する移動量
オフセット距離（Z 軸）	DELTA_Z	[数値] デフォルト： 0.0	Z 軸方向に適用する移動量
オフセット距離（M 値）	DELTA_M	[数値] デフォルト： 0.0	M 値に適用する移動量

次のページに続く

表 27.167 – 前のページからの続き

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力ベクタレイヤ

Python コード

Algorithm ID: `native:translategeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

可変幅バッファ（M 値使用）

ラインジオメトリの M 値を各頂点におけるバッファの直径として使用した、ラインに沿って幅が変化するバッファを作成します。

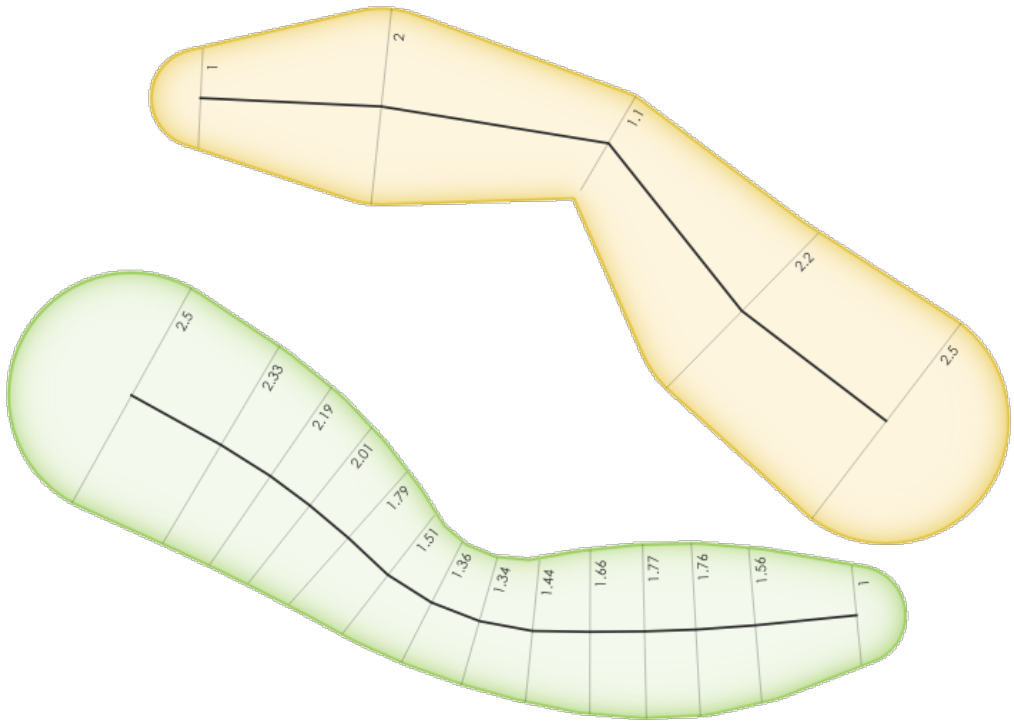


図 27.107: 可変幅バッファの例

参考:

テイパードバッファ, **バッファ（*buffer*）**, ***M* 値の設定**, **可変距離バッファ**

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインベクタレイヤ
セグメント	SEGMENTS	[数値] デフォルト： 16	バッファの四分円のセグメント数。単一の値（すべての地物に対して同じ値）とすることもできますし、地物データから取得して設定する（値は地物の属性に依存する）こともできます。
出力レイヤ	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（バッファの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
出力レイヤ	OUTPUT	[ベクタ：ポリゴン]	可変幅バッファのポリゴンレイヤ

Python コード

Algorithm ID: native:bufferbym

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ボロノイ多角形

ポイントレイヤを入力として、入力ポイントに対応するボロノイポリゴン（ティーセンポリゴンとも呼ばれる）のポリゴンレイヤを作成します。

あるボロノイポリゴン内の任意の地点から最も近いポイントは、そのポリゴンに関連付けられたポイントです。

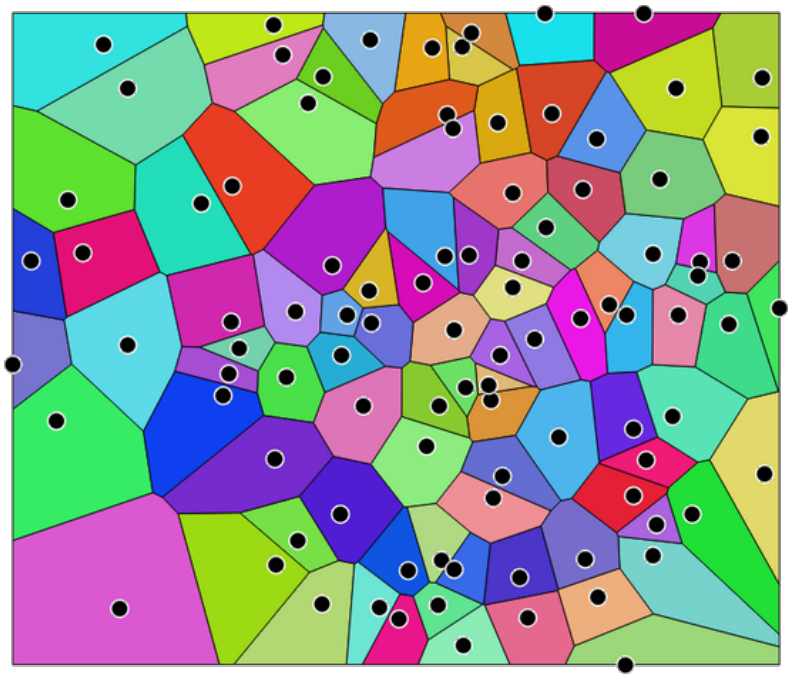


図 27.108: ボロノイ多角形

デフォルトメニュー ： ベクタ　ジオメトリツール

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
バッファ領域(領域に対する%単位)	BUFFER	[数値] デフォルト： 0.0	出力レイヤの範囲は入力レイヤの範囲よりもこの分だけ大きくなります
ボロノイ多角形	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時レイヤを作成]	（ボロノイポリゴンの）出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	データ型	説明
ボロノイ多角形	OUTPUT	[ベクタ：ポリゴン]	入力ポイントベクタレイヤに対するボロノイポリゴン

Python コード

Algorithm ID: `qgis:voronoipolygons`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.19 ベクタオーバーレイ

切り抜く

追加したポリゴンレイヤの地物を使ってベクタレイヤを切り抜きます。

入力レイヤの地物のうち、オーバーレイレイヤのポリゴンに含まれる部分のみが、結果レイヤに追加されます。

警告: ジオメトリの修正のみ

この操作は地物のジオメトリのみを変更します。地物の属性値は **変更されません** が、地物の面積や長さなどのプロパティはオーバーレイ操作によって変更されます。そのようなプロパティが属性として保存されている場合、それらの属性は手動で更新する必要があります。

このアルゴリズムは、プロバイダで空間インデックスを使用し、ジオメトリが準備されたジオメトリを使用し、ジオメトリがマスクジオメトリに完全に含まれていない場合に切り抜き操作を適用します。

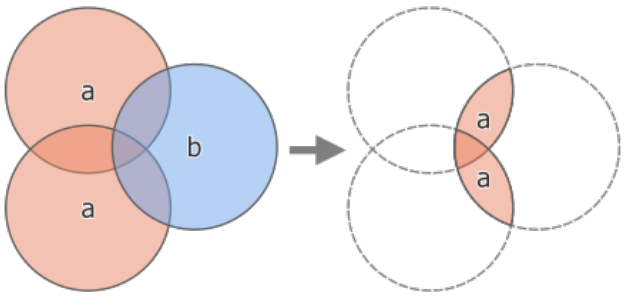


図 27.109: 2 つの地物を持つ入力レイヤ 'a' と 1 つの地物を持つオーバーレイレイヤ 'b' 間の切り抜き操作（左）- 結果として、修正された 'a' 地物を持つ新しいレイヤができる（右）

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー ： ベクタ　空間演算ツール

参考:

交差, **差分**

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	切り抜かれる地物を格納しているレイヤ
オーバーレイレイヤ	OVERLAY	[ベクタ：ポリゴン]	切り抜く地物を格納しているレイヤ
切り抜き結果	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	オーバーレイ（クリッピング）レイヤの内側にある入力レイヤの地物を格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
切り抜き結果	OUTPUT	[入力レイヤと同じ]	オーバーレイレイヤで分割された入力レイヤの地物を格納するレイヤ

Python コード

Algorithm ID: `qgis:clip`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

差分

入力レイヤから、オーバーレイレイヤの境界内にない地物を抽出します。

オーバーレイレイヤ地物と部分的に重なる入力レイヤ地物は、それらの地物の境界に沿って分割され、オーバーレイレイヤ地物の外側の部分のみが保持されます。

警告: ジオメトリの修正のみ

この操作は地物のジオメトリのみを変更します。地物の属性値は **変更されません** が、地物の面積や長さなどのプロパティはオーバーレイ操作によって変更されます。そのようなプロパティが属性として保存されている場合、それらの属性は手動で更新する必要があります。

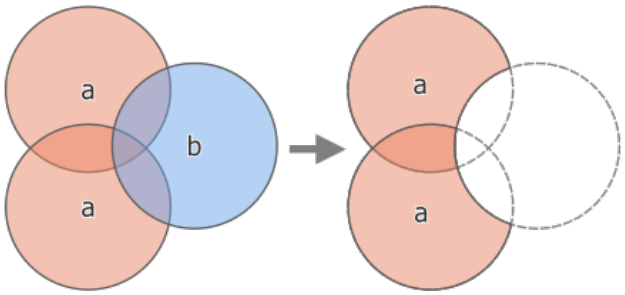


図 27.110: 2 つの地物を持つ入力レイヤ 'a' と 1 つの地物を持つオーバーレイレイヤ 'b' の差分操作（左） - 結果として、修正された 'a' 地物を持つ新しいレイヤ（右）

ポイント、ライン、ポリゴン地物の **地物の** *In-place* **編集** が可能です

デフォルトメニュー : **ベクタ　空間演算ツール**

参考:

差分（複数レイヤ）, **対称差**, **切り抜く**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物（の一部）を抽出するレイヤ。
オーバーレイレイヤ	OVERLAY	[ベクタ：任意]	入力レイヤのジオメトリから減算されるジオメトリを含むレイヤ。少なくとも入力レイヤのジオメトリと同じ次元数（ポイント：0D、ライン：1D、ポリゴン：2D、ボリューム：3D）を持つことが期待される。

次のページに続く

表 27.168 – 前のページからの続き

ラベル	名前	タイプ	説明
差分	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	入力レイヤの地物（の一部）のうち、オーバーレイレイヤの内側にないものを格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	タイプ	説明
グリッドサイズ NEW in 3.28 オプション	GRID_SIZE	[数値] デフォルト：未設定	指定された場合、入力ジオメトリは指定されたサイズのグリッドにスナップされ、結果の頂点は同じグリッド上で計算されます。GEOS 3.9.0 以上が必要です。

出力

ラベル	名前	タイプ	説明
差分	OUTPUT	[入力レイヤと同じ]	オーバーレイレイヤに重ならない、入力レイヤの地物（の一部）を格納するレイヤ。

Python コード

Algorithm ID: `qgis:difference`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

差分（複数レイヤ）

NEW in 3.26

入力レイヤから、いずれのオーバーレイレイヤの地物からも完全に外れているか部分的にしか重ならない、地物を抽出します。

各オーバーレイレイヤについて、それまでの全ての差分演算の結果とこのオーバーレイレイヤの差分が計算されます。オーバーレイレイヤの地物と部分的に重なる入力レイヤの地物は、それらの地物の境界に沿って分割され、オーバーレイレイヤの地物の外側の部分のみが保持されます。

警告: ジオメトリの修正のみ

この操作は地物のジオメトリのみを変更します。地物の属性値は **変更されません** が、地物の面積や長さなどのプロパティはオーバーレイ操作によって変更されます。そのようなプロパティが属性として保存されている場合、それらの属性は手動で更新する必要があります。

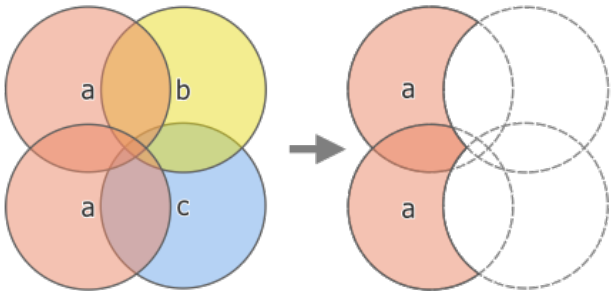


図 27.111: 2 つの地物を持つ入力レイヤ'a' と 1 つの地物を持つオーバーレイレイヤ'b' と'c' の差分演算 (左) - 結果として、修正された'a' 地物を持つ新しいレイヤ (右)

参考:

差分, 対称差, 切り抜く

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物（の一部）を抽出するレイヤ。
オーバーレイレイヤ	OVERLAYS	[ベクタ：任意] [リスト]	入力レイヤのジオメトリから減算されるジオメトリを格納しているレイヤのリスト。少なくとも入力レイヤのジオメトリと同じ次元（ポイント: 0D、ライン: 1D、ポリゴン: 2D、ボリューム: 3D）を持つことが期待されます。
差分	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	入力レイヤの地物（の一部）のうち、オーバーレイレイヤの地物と重ならないものを格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
差分	OUTPUT	[入力レイヤと同じ]	入力レイヤの地物（の一部）のうち、オーバーレイレイヤの地物と重ならないものを格納するレイヤ。

Python コード

Algorithm ID: `qgis:multidifference`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

矩形範囲で抽出

指定した範囲内にある地物のみを含む新しいベクタレイヤを作成します。

範囲と交差する地物は全て含まれます。

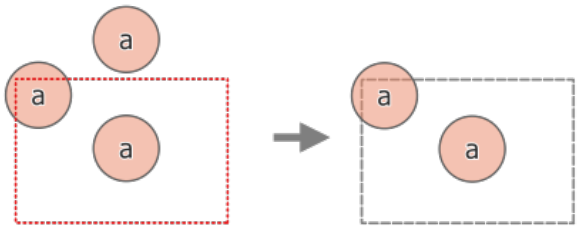
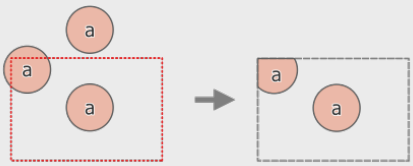


図 27.112: 3 つの地物を持つ入力レイヤ'a' と破線の範囲との間の抽出演算（左） - 結果の地物と参考の破線の範囲（右）

参考:

切り抜く

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物（の一部）を抽出するレイヤ。
領域 (xmin, xmax, ymin, ymax)	EXTENT	[範囲]	切り抜く範囲 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
地物を切り抜く	CLIP	[ブール値] デフォルト：False	チェックを付けると、出力ジオメトリは自動的にマルチジオメトリに変換され、出力タイプが均一になります。さらに、それらのジオメトリは、ジオメトリ全体を出力として使用するのではなく、選択した範囲で切り取られます。  図 27.113: 3 つの地物を持つ入力レイヤ 'a' と破線の範囲との間の抽出演算（左）- 結果の地物と参考の破線の範囲（右）
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	切り抜き範囲内にある入力レイヤの地物を格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	切り抜かれた地物を格納するレイヤ

Python コード

Algorithm ID: `qgis:extractbyextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

交差

入力レイヤの地物のうち、オーバーレイレイヤの地物と重なる部分を抽出します。

交差レイヤの地物には、入力レイヤとオーバーレイレイヤの両方から重なる地物の属性が割り当てられます。

警告: ジオメトリの修正のみ

この操作は地物のジオメトリのみを変更します。地物の属性値は **変更されません** が、地物の面積や長さなどのプロパティはオーバーレイ操作によって変更されます。そのようなプロパティが属性として保存されている場合、それらの属性は手動で更新する必要があります。

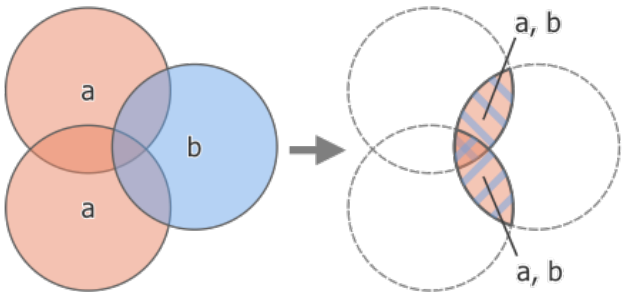


図 27.114: 2 つの地物を持つ入力レイヤ'a' とひとつの地物を持つオーバーレイレイヤ'b' の交差演算（左） - 重なり合った領域は、両方のレイヤの属性を持った、2 つの地物を持つ新しいレイヤとなる（右）

デフォルトメニュー ： ベクタ　空間演算ツール

参考:

交差（複数レイヤ）, **切り抜く**, **差分**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物（の一部）を抽出するレイヤ。
オーバーレイレイヤ	OVERLAY	[ベクタ：任意]	重なりをチェックする地物を格納しているレイヤ。その地物のジオメトリは、少なくとも入力レイヤと同じ次元（ポイント: 0D、ライン: 1D、ポリゴン: 2D、ボリューム: 3D）であることが期待されます。
入力レイヤからコピーする属性（全属性を保持する場合は空白のまま） オプション	INPUT_FIELDS	[テーブルのフィールド：任意] [リスト] デフォルト： なし	出力に残す入力レイヤのフィールド。フィールドが選択されないときは、全てのフィールドが出力されます。
オーバーレイレイヤからコピーする属性（全属性を保持する場合は空白のまま） オプション	OVERLAY_FIELDS	[テーブルのフィールド：任意] [リスト] デフォルト： なし	出力に残すオーバーレイレイヤのフィールド。フィールドが選択されないときは、全てのフィールドが出力されます。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。
交差	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	オーバーレイレイヤのひとつ以上の地物と重なる入力レイヤの地物（の一部）を格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	タイプ	説明
オーバーレイの属性の接頭辞 オプション	OVERLAY_FIELDS_P	[文字列]	オーバーレイレイヤのフィールドを識別するための接頭辞を追加します。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。
グリッドサイズ NEW in 3.28 オプション	GRID_SIZE	[数値] デフォルト：未設定	指定された場合、入力ジオメトリは指定されたサイズのグリッドにスナップされ、結果の頂点は同じグリッド上で計算されます。GEOS 3.9.0 以上が必要です。

出力

ラベル	名前	タイプ	説明
交差	OUTPUT	[入力レイヤと同じ]	オーバーレイレイヤと重なる、入力レイヤの地物（の部分）を格納するレイヤ。

Python コード

Algorithm ID: `qgis:intersection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

交差（複数レイヤ）

NEW in 3.26

入力レイヤ及び全てのオーバーレイレイヤの地物のうち、重なっている部分を抽出します。

出力レイヤの地物には、入力レイヤとオーバーレイレイヤの両方から重なる地物の属性が割り当てられます。

警告: ジオメトリの修正のみ

この操作は地物のジオメトリのみを変更します。地物の属性値は **変更されません** が、地物の面積や長さなどのプロパティはオーバーレイ操作によって変更されます。そのようなプロパティが属性として保存されている場合、それらの属性は手動で更新する必要があります。

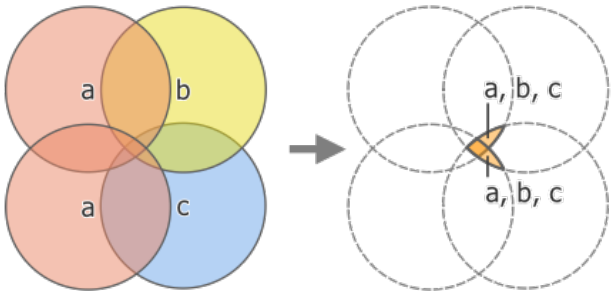


図 27.115: 2 つの地物を持つ入力レイヤ'a' とひとつの地物を持つオーバーレイレイヤ'b' と'c' の交差演算（左）- 重なり合った領域は、全てのレイヤの属性を持った、2 つの地物を持つ新しいレイヤとなる（右）

参考:

交差, **切り抜く**, **差分**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物（の一部）を抽出するレイヤ。
オーバーレイレイヤ	OVERLAYS	[ベクタ：任意] [リスト]	重なりをチェックする地物を格納しているレイヤ。その地物のジオメトリは、少なくとも入力レイヤと同じ次元（ポイント: 0D、ライン: 1D、ポリゴン: 2D、ボリューム: 3D）であることが期待されます。

詳細パラメータ

ラベル	名前	タイプ	説明
オーバーレイの属性の接頭辞 オプション	OVERLAY_FIELDS_P	[文字列]	オーバーレイレイヤのフィールドを識別するための接頭辞を追加します。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。

出力

ラベル	名前	タイプ	説明
交差	OUTPUT	[入力レイヤと同じ]	全てのオーバーレイレイヤと重なる、入力レイヤの地物（の部分）を格納するレイヤ。

Python コード

Algorithm ID: `qgis:multiintersection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線の交差

2 つのレイヤのラインが交差するポイント地物を作成します。

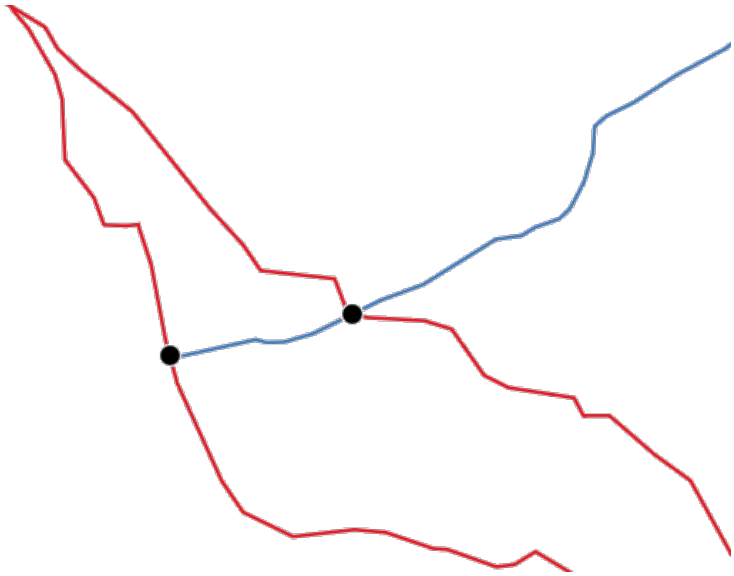


図 27.116: 交差の点

デフォルトメニュー: **ベクタ　解析ツール**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインレイヤ
交差レイヤ	INTERSECT	[ベクタ：ライン]	線の交差を見つけるために使うレイヤ。
入力レイヤからコピーする属性（全属性を保持する場合は空白のまま） オプション	INPUT_FIELDS	[テーブルのフィールド：任意] [リスト] デフォルト：なし	出力に残す入力レイヤのフィールド。フィールドが選択されないときは、全てのフィールドが出力されます。
交差レイヤからコピーする属性（全属性を保持する場合は空白のまま） オプション	INTERSECT_FIELDS	[テーブルのフィールド：任意] [リスト] デフォルト：なし	出力に残す交差レイヤのフィールド。フィールドが選択されないときは、全てのフィールドが出力されます。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。

次のページに続く

表 27.176 – 前のページからの続き

ラベル	名前	タイプ	説明
交差	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	入力レイヤとオーバーレイレイヤのラインの交差ポイントを格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	タイプ	説明
交差のフィールド名接頭辞 オプション	INTERSECT_FIELDS	[文字列]	交差レイヤのフィールドを識別するための接頭辞を追加します。

出力

ラベル	名前	タイプ	説明
交差部分	OUTPUT	[ベクタ：ポイント]	両方のレイヤの属性を持った、線の交差のポイントベクタレイヤ。

Python コード

Algorithm ID: qgis:lineintersections

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線で分割

あるレイヤのライン又はポリゴンを、別のレイヤのライン又はポリゴンリングを使って分割し、分割点を定義します。両方のレイヤのジオメトリ間の交点が分割点とみなされます。

出力は、分割された地物のマルチジオメトリを格納します。

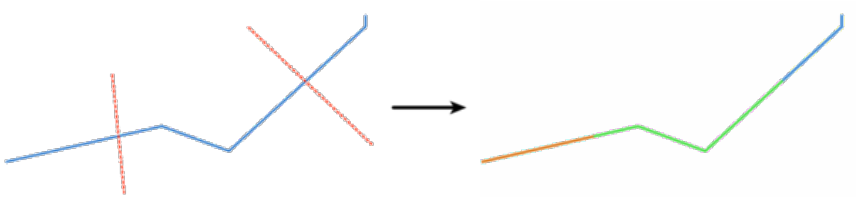


図 27.117: 分割されたライン

ライン及びポリゴン地物の **地物の** *In-place* **編集** が可能です

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ:ライン、ポリゴン]	分割されるライン又はポリゴンを格納しているレイヤ。
切断線のレイヤ	LINES	[ベクタ:ライン、ポリゴン]	分割点を定義するのに使われるライン又はポリゴンを持つレイヤ。
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	入力レイヤから分割された（分割レイヤにあるラインと交差した）ライン／ポリゴン地物を格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	入力レイヤから分割されたライン又はポリゴンを持つ出力ベクタレイヤ

Python コード

Algorithm ID: `qgis:splitwithlines`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

対称差

入力レイヤとオーバーレイレイヤの両方からの地物を含み、2 つのレイヤ間の重なっている領域が取り除かれたレイヤを作成します。

対称差レイヤの属性テーブルには、入力レイヤとオーバーレイレイヤの両方の属性とフィールドが含まれています。

警告: ジオメトリの修正のみ

この操作は地物のジオメトリのみを変更します。地物の属性値は **変更されません** が、地物の面積や長さなどのプロパティはオーバーレイ操作によって変更されます。そのようなプロパティが属性として保存されている場合、それらの属性は手動で更新する必要があります。

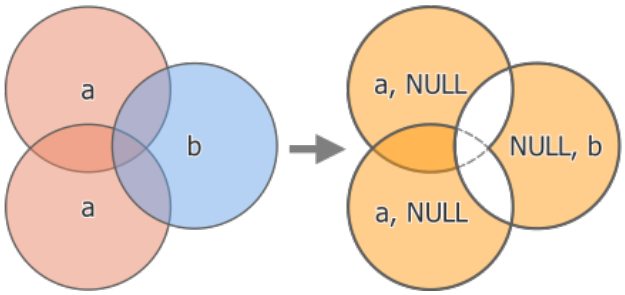


図 27.118: 2 つの地物を持つ入力レイヤ'a' とひとつの地物を持つオーバーレイレイヤ'b' の対称差演算（左）- 両方のレイヤの属性を持った、3 つの地物を持つレイヤとなる（右）

デフォルトメニュー ： ベクタ　 空間演算ツール

参考:

差分, **切り抜く**, **交差**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物（の部分）を抽出する最初のレイヤ。
オーバーレイレイヤ	OVERLAY	[ベクタ：任意]	地物（の部分）を抽出する二番目のレイヤ。ジオメトリの型は入力レイヤと同じであることが理想的です。
対象差	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	入力レイヤとオーバーレイレイヤからの地物（の部分）であって、他のレイヤの地物と重ならないものを格納するレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	タイプ	説明
オーバーレイの属性の接頭辞 オプション	OVERLAY_FIELDS_P	[文字列]	オーバーレイレイヤのフィールドを識別するための接頭辞を追加します。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。
グリッドサイズ NEW in 3.28 オプション	GRID_SIZE	[数値] デフォルト：未設定	指定された場合、入力ジオメトリは指定されたサイズのグリッドにスナップされ、結果の頂点は同じグリッド上で計算されます。GEOS 3.9.0 以上が必要です。

出力

ラベル	名前	タイプ	説明
対象差	OUTPUT	[入力レイヤと同じ]	各レイヤからの地物（の部分）であり、他のレイヤと重ならず、両方のレイヤの属性を持つものを格納するレイヤ。

Python コード

Algorithm ID: `qgis:symmetricaldifference`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

和集合

入力レイヤの地物が互いに重なっていないかどうかをチェックし、重なった部分と重なっていない部分を別の地物に分解します。重なった部分は、重なった地物の数だけ同じ地物が作成されます。

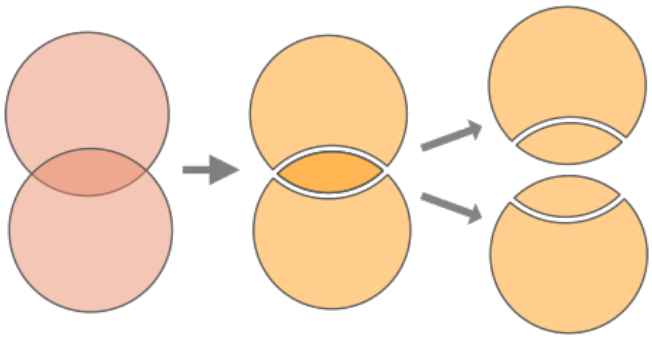


図 27.119: 1 つの入力レイヤで 2 つの地物が重なった和集合演算（左） - 4 つの地物になる（中） - 分かりやすくするために地物を移動（右）

オーバーレイレイヤを使うこともでき、この場合、各レイヤの地物は、もう一方のレイヤの地物と重なる部分で分割され、入力レイヤとオーバーレイレイヤの全ての部分を含むレイヤが作成されます。同じレイヤの地物は互いに分割されません。和集合レイヤの属性テーブルには、重複していない地物の場合はそれぞれの元のレイヤの属性値が、重複している地物の場合は両方のレイヤの属性値が入力されます。

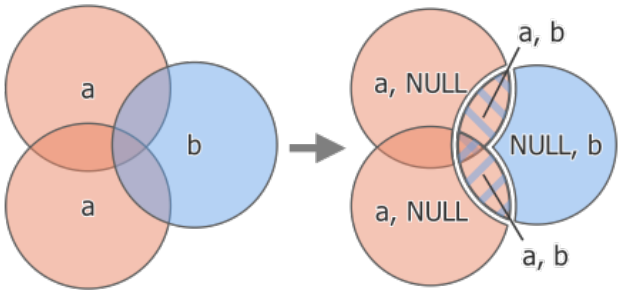


図 27.120: 2 地物の入力レイヤ「a」と 1 地物のオーバーレイレイヤ「b」の和集合演算（左） - 両レイヤの属性を持つ 5 地物のレイヤとなる（右）

注釈: オーバーレイレイヤでは、同じレイヤ上の地物は互いに分割されません。他のレイヤとの重なりだけでなく同じレイヤの重なりを分割したい場合は、まず複数のレイヤでアルゴリズムを実行し、前回の出力のみで再度アルゴリズムを実行してください。

デフォルトメニュー ： ベクタ　空間演算ツール

参考:

和集合（複数レイヤ）, **切り抜く**, **差分**, **交差**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	全ての交差で分割する入力ベクタレイヤ。
オーバーレイレイヤ オプション	OVERLAY	[ベクタ：任意]	最初のレイヤと結合されるレイヤ。ジオメトリタイプは入力レイヤと同じであることが理想的です。
和集合	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	入力レイヤおよびオーバーレイレイヤの（分割および複製された）地物を格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	タイプ	説明
オーバーレイの属性の接頭辞 オプション	OVERLAY_FIELDS_P	[文字列]	オーバーレイレイヤのフィールドを識別するための接頭辞を追加します。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。
グリッドサイズ NEW in 3.28 オプション	GRID_SIZE	[数値] デフォルト：未設定	指定された場合、入力ジオメトリは指定されたサイズのグリッドにスナップされ、結果の頂点は同じグリッド上で計算されます。GEOS 3.9.0 以上が必要です。

出力

ラベル	名前	タイプ	説明
和集合	OUTPUT	[入力レイヤと同じ]	処理されたレイヤの重複している部分と重複していない部分を全て格納するレイヤ。

Python コード

Algorithm ID: qgis:union

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

和集合（複数レイヤ）

NEW in 3.26

入力レイヤの地物が互いに重なっていないかどうかをチェックし、重なった部分と重なっていない部分を別の地物に分解します。重なった部分は、重なった地物の数だけ同じ地物が作成されます。

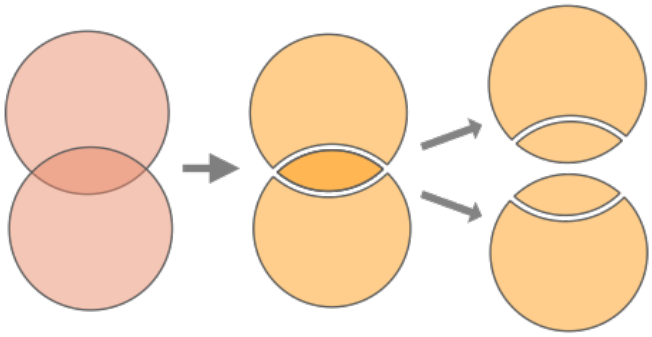


図 27.121: 1 つの入力レイヤで 2 つの地物が重なった和集合演算（左） - 4 つの地物になる（中） - 分かりやすくするために地物を移動（右）

複数のオーバーレイレイヤを使うこともできます、 この場合、各レイヤの地物は、他の全てのレイヤの地物と重なる部分で分割され、入力レイヤとオーバーレイレイヤの全ての部分を含むレイヤが作成されます。同じレイヤの地物は互いに分割されません。和集合レイヤの属性テーブルには、重複していない地物の場合はそれぞれの元のレイヤの属性値が、重複している地物の場合はオーバーレイレイヤの属性値が入力されます。

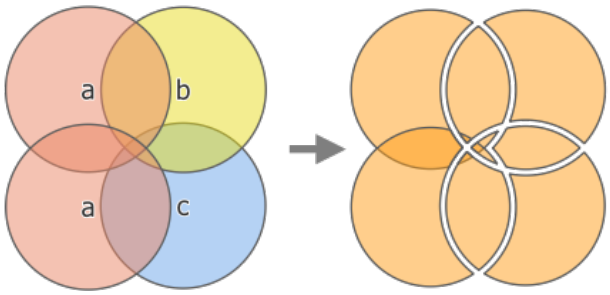


図 27.122: 2 地物の入力レイヤ「a」と 1 地物のオーバーレイレイヤ「b」及び「c」の和集合演算（左） - 全レイヤの属性を持つ 11 地物のレイヤとなる（右）

注釈: オーバーレイレイヤでは、同じレイヤ上の地物は互いに分割されません。他のレイヤとの重なりだけでなく同じレイヤの重なりを分割したい場合は、まず複数のレイヤでアルゴリズムを実行し、前回の出力のみで再度アルゴリズムを実行してください。

参考:

和集合, **切り抜く**, **差分**, **交差**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	全ての交差で分割する入力ベクタレイヤ。
オーバーレイレイヤ オプション	OVERLAYS	[ベクタ：任意] [リスト]	最初のレイヤと結合されるレイヤ。ジオメトリタイプは入力レイヤと同じであることが理想的です。
和集合	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	入力レイヤおよびオーバーレイレイヤの（分割および複製された）地物を格納するレイヤを指定します。以下のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

詳細パラメータ

ラベル	名前	タイプ	説明
オーバーレイの属性の接頭辞 オプション	OVERLAY_FIELDS_P	[文字列]	オーバーレイレイヤのフィールドを識別するための接頭辞を追加します。重複するフィールド名には、衝突を避けるためにカウントサフィックスが付加されます。

出力

ラベル	名前	タイプ	説明
和集合	OUTPUT	[入力レイヤと同じ]	処理されたレイヤの重複している部分と重複していない部分の全てを、全てのレイヤの属性と共に、格納するレイヤ。

Python コード

Algorithm ID: `qgis:multiunion`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.20 ベクタ選択

属性による抽出

入力レイヤから二つのベクタレイヤを作成する: ひとつは合致した地物だけを含み、もうひとつは合致しなかった全ての地物を含みます。

結果レイヤに地物を追加する規則は、入力レイヤからの属性の値に基づきます。

参考:

属性による選択

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物を抽出するレイヤ。
選択基準になる属性	FIELD	[テーブルのフィールド：任意]	レイヤをフィルタするフィールド
演算子	OPERATOR	[列挙型] デフォルト： 0	たくさんの種類の演算子が利用できます: • 0 --- = • 1 --- ≠ • 2 --- >
• 3 --- >= • 4 --- <
• 5 --- <= • 6 --- で始まる • 7 -- 含む • 8 --- null である • 9 --- null ではない • 10 --- 含まれていない

次のページに続く

表 27.181 – 前のページからの続き

ラベル	名前	タイプ	説明
値 オプション	VALUE	[文字列]	評価される値
抽出された属性	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	合致した地物の出力ベクタレイヤを指定します。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
抽出された属性（不一致）	FAIL_OUTPUT	[入力レイヤと同じ] デフォルト：[出力をスキップ]	合致しない地物の出力ベクタレイヤを指定します。次のいずれかです: • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
抽出された属性	OUTPUT	[入力レイヤと同じ]	入力レイヤから作られた、適合した地物を持つベクタレイヤ
抽出された属性（不一致）	FAIL_OUTPUT	[入力レイヤと同じ]	入力レイヤから作られた、適合しない地物を持つベクタレイヤ

Python コード

Algorithm ID: `qgis:extractbyattribute`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

QGIS 式による抽出

入力レイヤから二つのベクタレイヤを作成する: ひとつは合致した地物だけを含み、もうひとつは合致しなかった全ての地物を含みます。

結果レイヤに地物を追加する基準は、QGIS 式に基づいています。式の詳細については **式** を参照してください。

参考:

QGIS **式による選択**

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
式	EXPRESSION	[式]	ベクタレイヤをフィルタする式
出力レイヤ（マッチした地物）	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	合致した地物の出力ベクタレイヤを指定します。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
マッチしなかった地物	FAIL_OUTPUT	[入力レイヤと同じ] デフォルト：[出力をスキップ]	合致しない地物の出力ベクタレイヤを指定します。次のいずれかです: • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ（マッチした地物）	OUTPUT	[入力レイヤと同じ]	入力レイヤから作られた、適合した地物を持つベクタレイヤ
マッチしなかった地物	FAIL_OUTPUT	[入力レイヤと同じ]	入力レイヤから作られた、適合しない地物を持つベクタレイヤ

Python コード

Algorithm ID: `qgis:extractbyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

場所による抽出

入力レイヤからマッチする地物だけを含んだ新しいベクタレイヤを作ります。

結果レイヤに地物を追加する基準は、各地物と追加レイヤにある地物の空間的関係に基づきます。

参考:

場所による選択, **一定距離以内の地物を抽出**

空間的関係を調べる

幾何学的述語は、ある地物が他の地物と空間の一部を共有しているかどうか、またどのように共有しているかを比較することによって、空間的関係を持つかどうかを決定するために使用されるブール関数です。

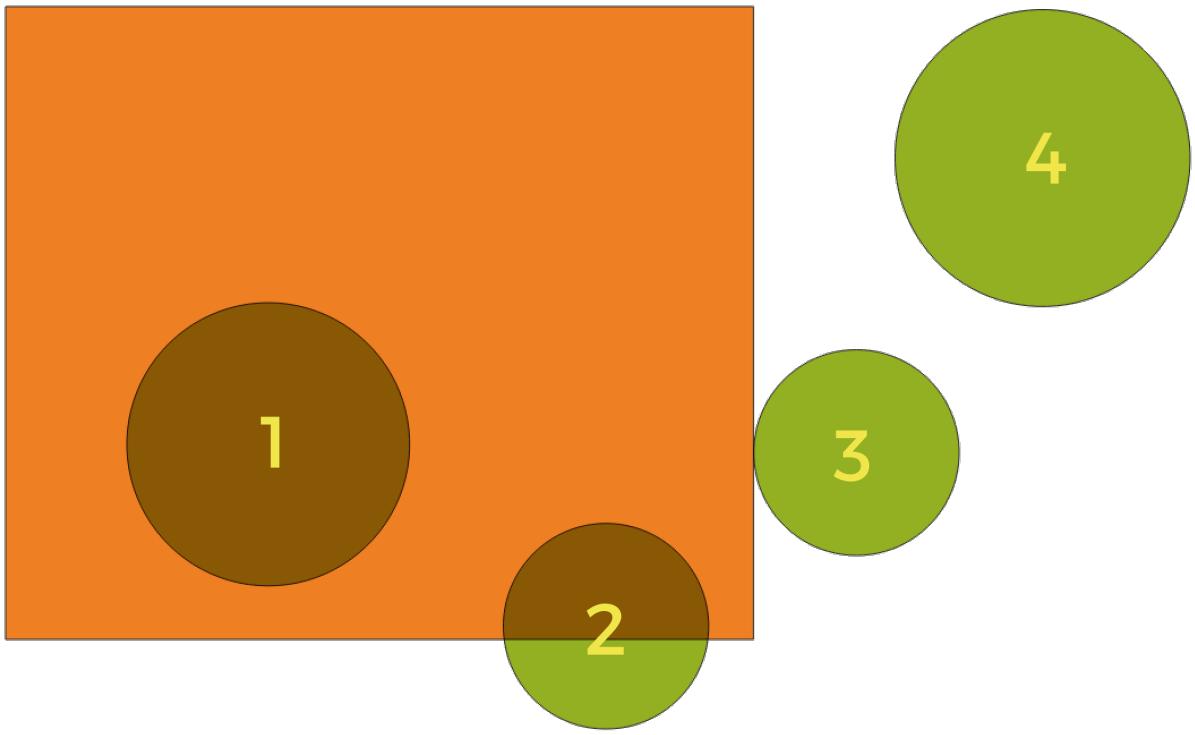


図 27.123: レイヤ間の空間的関係を求める

上の図を使って、オレンジ色の長方形の地物と空間的に比較することで、緑色の円を求めます。利用可能な幾何学的述語は以下の通りです：

交差する ***(intersect)***
: あるジオメトリが他のジオメトリと交差しているかどうかをテストします。ジオメトリが空間的に交差している（空間の一部を共有している - 重なり合うか接触している）場合は 1（true）を返し、交差していない場合は 0 を返します。上の図では、円 1、2、3 を返します。

含む ***(contain)***
: b の点が a の外側になく、かつ b の内側の少なくとも 1 点が a の内側にある場合にのみ、1 (true) を返します。この図では、どの円も返されませんが、逆を探せば、長方形は円 1 を完全に含むので、返されます。これは **含まれる** *(within)* の逆です。

離れている ***(disjoint)***
: ジオメトリが空間のどの部分も共有していない（重なっていない、接触していない）場合、1 (true) を返します。円 4 のみが返されます。

等しい ***(equal)***
: ジオメトリが完全に同じ場合のみ、1 (true) を返します。どの円も返されません。

接触する ***(touch)***
: あるジオメトリが別のジオメトリに接しているかどうかを調べます。ジオメトリに少なくとも 1 つの共通点があるが、内部が交差していない場合は 1 (true) を返します。円 3 のみが返されます。

重なる ***(overtap)***
: あるジオメトリが別のジオメトリに重なっているかどうかを調べます。ジオメトリが空間を共有し、同じ次元であるが、互いに完全に含まれない場合は 1 (true) を返します。円 2 のみが返されます。

含まれる *(within)*

あるジオメトリが他のジオメトリの中にあるかどうかを調べます。ジオメトリ a が完全にジオメトリ b の内側にある場合は 1 (true) を返します。

交差する *(cross)*

与えられたジオメトリが、すべてではなく、いくつかの内部点を共通に持ち、交差部が与えられたジオメトリの最大よりも小さい次元の場合、1 (true) を返します。例えば、ポリゴンを横切るラインは、ラインとして交差します (true)。交差する 2 本のラインは、ポイントとして交差します (true)。2 つのポリゴンはポリゴンとして交差します (false)。図では、どの円は返されません。

パラメーター

ラベル	名前	タイプ	説明
抽出する地物のあるレイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
空間的関係	PREDICATE	[列挙型] [リスト] デフォルト： [0]	入力地物が交差する地物との間に持っている場合に選択される、空間的関係の種類。次の 1 つ以上： • 0 -- 交差する (intersect) • 1 -- 含む (contain) • 2 -- 離れている (disjoint) • 3 -- 等しい (equal) • 4 -- 接触する (touch) • 5 -- 重なる (overlap) • 6 -- 含まれる (within) • 7 -- 交差する (cross) 複数の条件が選択された場合、そのうちの少なくとも 1 つ（OR 演算）を満たさなければ地物は抽出されません。
比較対象の地物のあるレイヤ	INTERSECT	[ベクタ：任意]	交差ベクタレイヤ
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	比較レイヤの 1 つ以上の地物と選択された空間的関係を持つ地物の出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	比較レイヤのひとつ以上の地物と、選ばれた空間的関係を持った、入力レイヤからの地物を持つベクタレイヤ。

Python コード

Algorithm ID: qgis:extractbylocation

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

一定距離以内の地物を抽出

入力レイヤからマッチする地物のみを含む新しいベクタレイヤを作成します。追加参照レイヤの地物から指定された最大距離以内にあるあらゆる地物がコピーされます。

参考:

一定距離以内の地物を選択, **場所による抽出**

パラメーター

ラベル	名前	タイプ	説明
抽出する地物のあるレイヤ	INPUT	[ベクタ：任意]	地物をコピーする入力ベクタレイヤ
比較対象の地物のあるレイヤ	REFERENCE	[ベクタ：任意]	その地物の近さを利用するベクタレイヤ
地物がある範囲	DISTANCE	[数値] デフォルト： 100	その範囲にある入力地物が選択される、参照地物からの最大距離

次のページに続く

表 27.183 – 前のページからの続き

ラベル	名前	タイプ	説明
現在の選択状態を以下のように変更する	METHOD	[列挙型] デフォルト： 0	アルゴリズムの選択がどのように処理されるか。次のいずれか: • 0 -- 新たに選択 • 1 -- 現在の選択に追加 • 2 -- 現在の選択の中から選択する • 3 -- 現在の選択から除く
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	参照地物から設定された距離以内にある地物の出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	参照地物からの距離の条件に合った入力レイヤの地物を持つベクタレイヤ

Python コード

Algorithm ID: native:extractwithindistance

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ジオメトリ型でフィルタ

地物をジオメトリタイプでフィルタします。入力された地物は、点、線、ポリゴンのどのジオメトリを持つかどうかによって異なる出力に導かれます。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	評価するレイヤ

出力

ラベル	名前	タイプ	説明
点地物 オプション	POINTS	[ベクタ：ポイント]	ポイントを持つレイヤ
線地物 オプション	LINES	[ベクタ：ライン]	ラインを持つレイヤ
ポリゴン地物 オプション	POLYGONS	[ベクタ：ポリゴン]	ポリゴンを持つレイヤ
ジオメトリのない地物 オプション	NO_GEOMETRY	[テーブル]	ジオメトリのないベクタレイヤ

Python コード

Algorithm ID: `native:filterbygeometry`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ランダム抽出

ベクタレイヤを取り、そのレイヤにある地物のサブセットだけを含んだ新しいベクタレイヤを生成します。

サブセットは地物 ID に基づいてランダムに決まり、サブセットに含まれる地物の数は百分率またはカウント値を使用して決まります。

参考:

ランダム選択

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物が選択されるソースベクタレイヤ
方法	METHOD	[列挙型] デフォルト： 0	ランダム選択の方法。次のいずれか： • 0 -- 地物の数 • 1 -- 地物のパーセンテージ
地物の数/パーセント	NUMBER	[数値] デフォルト： 10	選択する地物の数または百分率
ランダム抽出出力	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	ランダムに選択した地物の出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（ TEMPORARY_OUTPUT ） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
ランダム抽出出力	OUTPUT	[入力レイヤと同じ]	入力レイヤからランダムに選択された地物を含んだベクタレイヤ

Python コード

Algorithm ID: `qgis:randomextract`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地物のグループ別ランダム抽出

ベクタレイヤを取り、そのレイヤにある地物のサブセットだけを含んだ新しいベクタレイヤを生成します。

サブセットは地物 ID に基づいてランダムに決められ、サブセット内の地物の数は百分率またはカウント値を使って決められます。百分率/カウント値はレイヤ全体に適用されるのではなく、各カテゴリに適用されます。カテゴリは与えられた属性に従って定義されます。

参考:

地物のグループ別ランダム選択

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物が選択されるベクタレイヤ
グループの属性(フィールド)	FIELD	[テーブルのフィールド：任意]	地物が選択されるソースベクタレイヤのカテゴリ
方法	METHOD	[列挙型] デフォルト： 0	ランダム選択の方法。次のいずれか： • 0 -- 地物の数 • 1 -- 地物のパーセンテージ
地物の数/パーセント	NUMBER	[数値] デフォルト： 10	選択する地物の数または百分率

次のページに続く

表 27.185 – 前のページからの続き

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	ランダムに選択した地物の出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	入力レイヤからランダムに選択された地物を含んだベクタレイヤ

Python コード

Algorithm ID: qgis:randomextractwithinsubsets

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ランダム選択

ベクタレイヤを取り、その地物のサブセットを選択します。このアルゴリズムは新しいレイヤを生成しません。

サブセットは地物 ID に基づいてランダムに決まり、サブセットに含まれる地物の数は百分率またはカウント値を使用して決まります。

デフォルトメニュー ： ベクタ　 調査ツール

参考:

ランダム抽出

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	選択用のベクタレイヤ
方法	METHOD	[列挙型] デフォルト： 0	ランダム選択の方法。次のいずれか： • 0 -- 地物の数 • 1 -- 地物のパーセンテージ
地物の数/パーセント	NUMBER	[数値] デフォルト： 10	選択する地物の数または百分率

出力

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[入力レイヤと同じ]	地物が選択された入力レイヤ

Python コード

Algorithm ID: `qgis:randomselection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地物のグループ別ランダム選択

ベクタレイヤを取り、その地物のサブセットを選択します。このアルゴリズムは新しいレイヤを生成しません。

サブセットは地物 ID に基づいてランダムに決まり、サブセットに含まれる地物の数は百分率またはカウント値を使用して決まります。

百分率／カウント値はレイヤ全体ではなく、各カテゴリに適用されます。

カテゴリは、アルゴリズムの入力パラメータとしても指定される、与えられた属性に従って決められます。

新しい出力は生成されません。

デフォルトメニュー ： ベクタ　 調査ツール

参考:

地物のグループ別ランダム抽出

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物が選択されるベクタレイヤ
グループの属性(フィールド)	FIELD	[テーブルのフィールド：任意]	地物が選択される入力レイヤのカテゴリ
方法	METHOD	[列挙型] デフォルト： 0	ランダム選択の方法。次のいずれか： • 0 -- 地物の数 • 1 -- 地物のパーセンテージ
地物の数/パーセント	NUMBER	[数値] デフォルト： 10	選択する地物の数または百分率

出力

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[入力レイヤと同じ]	地物が選択された入力レイヤ

Python コード

Algorithm ID: `qgis:randomselectionwithinsubsets`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性による選択

ベクタレイヤに選択を生成します。

地物を選択する規則は、入力レイヤからの属性の値に基づきます。

参考:

属性による抽出

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	地物が選択されるベクタレイヤ
選択基準になる属性	FIELD	[テーブルのフィールド：任意]	レイヤをフィルタするフィールド
演算子	OPERATOR	[列挙型] デフォルト：0	たくさんの種類の演算子が利用できます: • 0 --- = • 1 --- ≠ • 2 --- >
• 3 --- >=
• 4 --- <
• 5 --- <=
• 6 --- で始まる
• 7 -- 含む
• 8 --- null である
• 9 --- null ではない
• 10 --- 含まれていない
値 オプション	VALUE	[文字列]	評価される値
現在の選択状態を以下のように変更する	METHOD	[列挙型] デフォルト：0	アルゴリズムの選択がどのように処理されるか。次のいずれか: • 0 -- 新たに選択
• 1 -- 現在の選択に追加
• 2 -- 現在の選択から除く
• 3 -- 現在の選択範囲内で選択しています

出力

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[入力レイヤと同じ]	地物が選択された入力レイヤ

Python コード

Algorithm ID: `qgis:selectbyattribute`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

QGIS 式による選択

ベクタレイヤに選択を生成します。

地物を選択する基準は QGIS 式に基づいています。式の詳細については **式** を参照してください。

参考:

QGIS **式による抽出**

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
式	EXPRESSION	[式]	入力レイヤをフィルタする式
現在の選択状態を以下のように変更する	METHOD	[列挙型] デフォルト： 0	アルゴリズムの選択がどのように処理されるか。次のいずれか: • 0 -- 新たに選択 • 1 -- 現在の選択に追加 • 2 -- 現在の選択から除く • 3 -- 現在の選択範囲内で選択しています

出力

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[入力レイヤと同じ]	地物が選択された入力レイヤ

Python コード

Algorithm ID: `qgis:selectbyexpression`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

場所による選択

ベクタレイヤに選択を生成します。

地物を選択する基準は、各地物と追加レイヤの地物との空間的関係に基づいています。

デフォルトメニュー : **ベクタ　調査ツール**

参考:

場所による抽出, **一定距離以内の地物を選択**

空間的関係を調べる

幾何学的述語は、ある地物が他の地物と空間の一部を共有しているかどうか、またどのように共有しているかを比較することによって、空間的関係を持つかどうかを決定するために使用されるブール関数です。

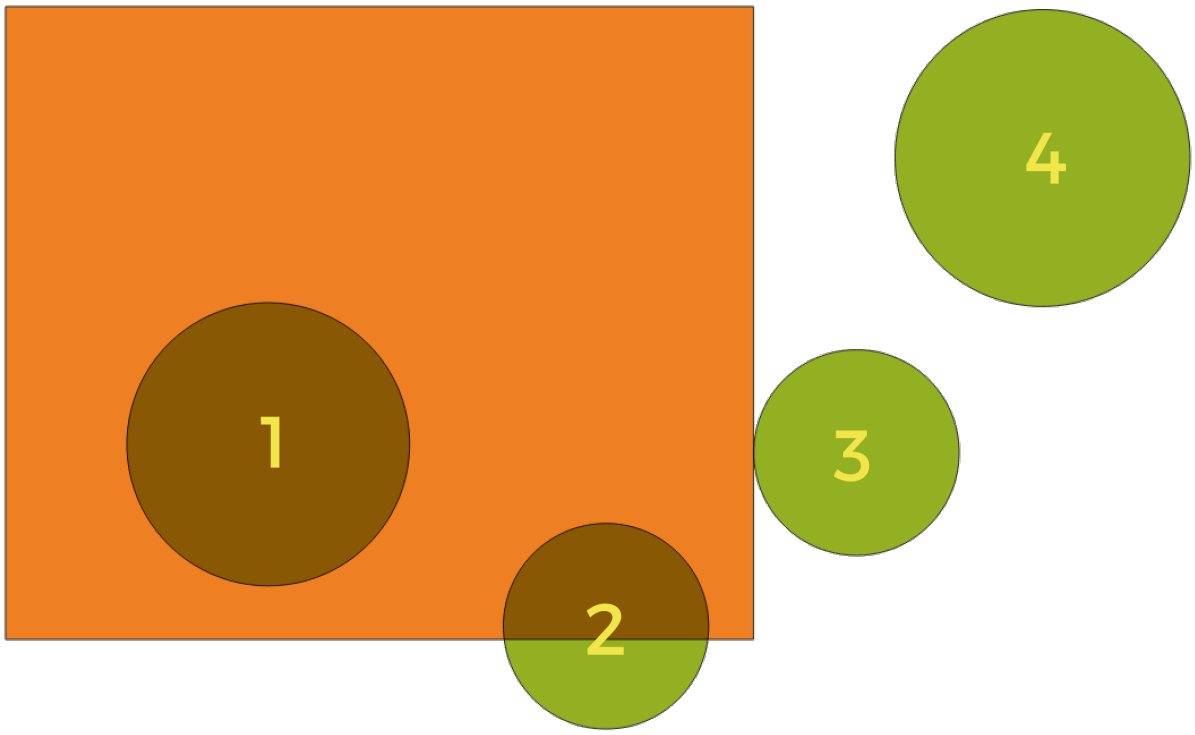


図 27.124: レイヤ間の空間的関係を求める

上の図を使って、オレンジ色の長方形の地物と空間的に比較することで、緑色の円を求めます。利用可能な幾何学的述語は以下の通りです：

交差する *(intersect)*
: あるジオメトリが他のジオメトリと交差しているかどうかをテストします。ジオメトリが空間的に交差している（空間の一部を共有している - 重なり合うか接触している）場合は 1（true）を返し、交差していない場合は 0 を返します。上の図では、円 1、2、3 を返します。

含む *(contain)*
: b の点が a の外側になく、かつ b の内側の少なくとも 1 点が a の内側にある場合にのみ、1 (true) を返します。この図では、どの円も返されませんが、逆を探せば、長方形は円 1 を完全に含むので、返されます。これは **含まれる** *(within)* の逆です。

離れている *(disjoint)*
: ジオメトリが空間のどの部分も共有していない（重なっていない、接触していない）場合、1 (true) を返します。円 4 のみが返されます。

等しい *(equal)*
: ジオメトリが完全に同じ場合のみ、1 (true) を返します。どの円も返されません。

接触する *(touch)*
: あるジオメトリが別のジオメトリに接しているかどうかを調べます。ジオメトリに少なくとも 1 つの共通点があるが、内部が交差していない場合は 1 (true) を返します。円 3 のみが返されます。

重なる *(overtap)*
: あるジオメトリが別のジオメトリに重なっているかどうかを調べます。ジオメトリが空間を共有し、同じ次元であるが、互いに完全に含まれない場合は 1 (true) を返します。円 2 のみが返されます。

含まれる *(within)*

あるジオメトリが他のジオメトリの中にあるかどうかを調べます。ジオメトリ a が完全にジオメトリ b の内側にある場合は 1 (true) を返します。

交差する *(cross)*

与えられたジオメトリが、すべてではなく、いくつかの内部点を共通に持ち、交差部が与えられたジオメトリの最大よりも小さい次元の場合、1 (true) を返します。例えば、ポリゴンを横切るラインは、ラインとして交差します (true)。交差する 2 本のラインは、ポイントとして交差します (true)。2 つのポリゴンはポリゴンとして交差します (false)。図では、どの円は返されません。

パラメーター

ラベル	名前	タイプ	説明
選択する地物のあるレイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
空間的関係	PREDICATE	[列挙型] [リスト] デフォルト： [0]	入力地物が交差する地物との間に持っている場合に選択される、空間的関係の種類。次の 1 つ以上： • 0 -- 交差する (intersect) • 1 -- 含む (contain) • 2 -- 離れている (disjoint) • 3 -- 等しい (equal) • 4 -- 接触する (touch) • 5 -- 重なる (overlap) • 6 -- 含まれる (within) • 7 -- 交差する (cross) 複数の条件が選択された場合、そのうちの少なくとも 1 つ（OR 演算）を満たさなければ地物は抽出されません。
比較対象の地物のあるレイヤ	INTERSECT	[ベクタ：任意]	交差ベクタレイヤ
現在の選択状態を以下のように変更する	METHOD	[列挙型] デフォルト： 0	アルゴリズムの選択がどのように処理されるか。次のいずれか: • 0 -- 新たに選択 • 1 -- 現在の選択に追加 • 2 -- 現在の選択の中から選択する • 3 -- 現在の選択から除く

出力

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[入力レイヤと同じ]	地物が選択された入力レイヤ

Python コード

Algorithm ID: `qgis:selectbylocation`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

一定距離以内の地物を選択

ベクタレイヤに選択を作ります。追加参照レイヤの地物から指定された最大距離以内にあるあらゆる地物が選択されます。

参考:

一定距離以内の地物を抽出, **場所による選択**

パラメーター

ラベル	名前	タイプ	説明
選択する地物のあるレイヤ	INPUT	[ベクタ：任意]	地物を選択する入力ベクタレイヤ
比較対象の地物のあるレイヤ	REFERENCE	[ベクタ：任意]	その地物の近さを利用するベクタレイヤ
地物がある範囲	DISTANCE	[数値] デフォルト： 100	入力地物が選択される、参照地物からの最大距離

次のページに続く

表 27.188 – 前のページからの続き

ラベル	名前	タイプ	説明
現在の選択状態を以下のように変更する	METHOD	[列挙型] デフォルト： 0	アルゴリズムの選択がどのように処理されるか。次のいずれか: • 0 -- 新たに選択 • 1 -- 現在の選択に追加 • 2 -- 現在の選択の中から選択する • 3 -- 現在の選択から除く

出力

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[入力レイヤと同じ]	地物が選択された入力レイヤ

Python コード

Algorithm ID: native:selectwithindistance

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.21 ベクタテーブル

自動インクリメント属性を追加

各地物に連続した値を持たせる新しい整数フィールドをベクタレイヤに追加します。

このフィールドは、レイヤ内の地物の一意の ID として使用できます。新しい属性は入力レイヤに追加されませんが、代わりに新しいレイヤが生成されます。

増加数列の初期開始値を指定できます。必要に応じて、増加数列はグループ化フィールドに基づくことができ、地物のソート順も指定できます。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
属性名	FIELD_NAME	[文字列] デフォルト：'AUTO'	自動増加値を格納するフィールド名
開始値 オプション	START	[数値] デフォルト：0	増加するカウントの最初の数を指定します
絶対値 (modulus) オプション	MODULUS	[数値] デフォルト：0	オプションで絶対値を指定すると、フィールドの値が絶対値に達するたびにカウントを START に戻します。0 は戻さないことを意味します。
グループ化のための属性 オプション	GROUP_FIELDS	[テーブルのフィールド：任意] [リスト]	グループ化フィールドを選択：レイヤー全体に対して 1 つのカウントを走らせる代わりに、これらのフィールドの組み合わせによって返される各値に対して別々のカウントを処理します。
ソート式 オプション	SORT_EXPRESSION	[式]	式を使い、レイヤの地物をグローバルに、または設定したグループフィールドに基づいて、ソートします。
昇順	SORT_ASCENDING	[ブール値] デフォルト：True	ソート式 が設定されているときに、このオプションを使用して地物に値が割り当てられる順番を制御します。
NULL は最初にソートされる	SORT_NULLS_FIRST	[ブール値] デフォルト：False	ソート式 が設定されているときに、このオプションを使用して *Null* 値を最初に数えるか最後に数えるかを設定します。
昇順	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	自動的に増加する属性を持った出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
昇順	OUTPUT	[入力レイヤと同じ]	自動的に増加する属性を持ったベクタレイヤ

Python コード

Algorithm ID: native:addautoincrementalfield

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性テーブルに属性を追加

新しいフィールドをベクタレイヤに追加します。

属性の名前と性質はパラメータで定義します。

新しい属性は入力レイヤに追加されず、代わりに新しいレイヤが生成されます。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力レイヤ
属性名	FIELD_NAME	[文字列]	新しいフィールドの名前
フィールド型	FIELD_TYPE	[列挙型] デフォルト： 0	新しいフィールドの型。次から選びます： • 0 --- 整数 • 1 --- Float • 2 --- 文字列
フィールド精度	FIELD_LENGTH	[数値] デフォルト： 10	フィールド長さ
フィールド精度	FIELD_PRECISION	[数値] デフォルト： 0	フィールドの精度。フィールド型が Float のときに有用です。

次のページに続く

表 27.190 – 前のページからの続き

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	新しいフィールドが追加されたベクタレイヤ

Python コード

Algorithm ID: `native:addfieldtoattributestable`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

分類用ユニーク属性を追加

ベクタレイヤとひとつの属性から新しい数値フィールドを追加します。

このフィールドの値は指定された属性の値に対応しているため、その属性に対して同じ値を持つ地物は新しい数値フィールドでも同じ値になります。

これは指定した属性に対して数値的に等価なものを生成し、分類を定義します。

新しい属性は入力レイヤに追加されず、代わりに新しいレイヤが生成されます。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力レイヤ
分類属性	FIELD	[テーブルのフィールド：任意]	このフィールドに同じ値を持つ地物は同じインデックスを得ます。
出力する属性（フィールド）の名前	FIELD_NAME	[文字列] デフォルト: 'NUM_FIELD'	インデックスを格納する新しいフィールドの名前。
出力レイヤ	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	インデックスを格納する数値フィールドを持つベクタレイヤ。次のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。
クラスの要約	SUMMARY_OUTPUT	[テーブル] デフォルト：[出力をスキップ]	対応する一意な値にマップされた分類フィールドの要約を格納するテーブルを指定します。以下のいずれかです： • 出力をスキップ • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	インデックスを格納した数値フィールドを持つベクタレイヤ。
クラスの要約	SUMMARY_OUTPUT	[テーブル]	対応する一意な値にマップされた分類フィールドの要約を持つテーブル。

Python コード

Algorithm ID: `native:adduniquevalueindexfield`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

X/Y フィールドを追加

ポイントレイヤに X と Y（または緯度／経度）フィールドを追加します。X/Y フィールドは、レイヤと異なる CRS で計算することができます（例えば、投影 CRS のレイヤで緯度／経度フィールドを作成します）。

ライン地物の **地物の *In-place* 編集** が可能です

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：ポイント]	入力レイヤ
座標系	CRS	[crs] デフォルト：EPSG:4326	生成する x と y フィールドに使用する座標参照系
属性名の接頭辞 オプション	PREFIX	[文字列]	入力レイヤのフィールドとの名前の衝突を避けるために、新しいフィールド名に加える接頭辞。
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	出力レイヤ - 2 つの新しい double のフィールド、x と y があるのを除けば入力レイヤと同じです。

Python コード

Algorithm ID: native:addxyfieldstolayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

Python フィールド計算機

各地物に式を適用して得られる値を持った、新しい属性をベクタレイヤに加えます。

式は Python 関数として定義されます。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
追加するフィールドの名前	FIELD_NAME	[文字列] デフォルト: 'New-Field'	新しいフィールドの名前
フィールド型	FIELD_TYPE	[列挙型] デフォルト： 0	新しいフィールドの型。次のいずれかです： • 0 --- 整数 • 1 --- Float • 2 --- 文字列
フィールド精度	FIELD_LENGTH	[数値] デフォルト： 10	フィールド長さ

次のページに続く

表 27.193 – 前のページからの続き

ラベル	名前	タイプ	説明
フィールド精度	FIELD_PRECISION	[数値] デフォルト： 3	フィールドの精度。フィールド型が Float のときに有用です。
グローバル式 オプション	GLOBAL	[文字列]	グローバル式セクションのコードは、計算機が入力レイヤのすべての地物を繰り返し処理する前に 1 回だけ実行されます。 したがって、これは、必要なモジュールをインポートしたり、以降の計算で使用される変数を計算したりするための正しい場所です。
計算式	FORMULA	[文字列]	評価する Python 計算式。例: 入力ポリゴンレイヤの面積を計算するには、次を追加します: `value = $geom.area()`
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	新しい計算したフィールドを持ったベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	新しい計算したフィールドを持ったベクタレイヤ

Python コード

Algorithm ID: `qgis:advancedpythonfieldcalculator`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性を削除

ベクタレイヤを取り、同じ地物を持ちますが、選択した列は持たない、新しいレイヤを生成します。

参考:

属性を選択保持

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	フィールドを削除する入力ベクタレイヤ
削除する属性（フィールド）	COLUMN	[テーブルのフィールド：任意] [リスト]	削除するフィールド
出力レイヤ	OUTPUT	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	残りのフィールドを持った出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	残りのフィールドを持ったベクタレイヤ

Python コード

Algorithm ID: `native:deletecolumn`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

HStore フィールドを展開

入力レイヤのコピーを作成し、HStore フィールドの一意キーごとに新しいフィールドを追加します。

期待されるフィールドリストは、オプションのカンマ区切りリストです。このリストを指定すると、これらのフィールドのみが追加され、HStore フィールドが更新されます。デフォルトでは、すべての一意キーが追加されます。

PostgreSQL の HStore は、PostgreSQL と GDAL （`other_tags` フィールドを持つ OSM ファイル を読み込むとき） で使用される、単純なキーと値のストアです。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	`INPUT`	[ベクタ：任意]	入力ベクタレイヤ
HStore フィールド	`FIELD`	[テーブルのフィールド：任意]	削除するフィールド
選択するフィールド（カンマ区切り） オプション	`EXPECTED_FIELDS`	[文字列] デフォルト：''	展開するフィールドのコンマ区切りのリスト。HStore フィールドはそれらのキーを削除することで更新されます。
出力レイヤ	`OUTPUT`	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力ベクタレイヤを指定します。次のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力ベクタレイヤ

Python コード

Algorithm ID: native:explodehstorefield

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

バイナリ属性を解凍

バイナリ・フィールドからコンテンツを抽出し、個々のファイルに保存します。ファイル名は、ソース・テーブルの属性から取得した値を使用して生成することも、より複雑な式に基づいて生成することもできます。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	バイナリデータを含んでいる入力ベクタレイヤ
バイナリ属性	FIELD	[テーブルのフィールド：任意]	バイナリデータのフィールド
ファイル名	FILENAME	[式]	各出力ファイルに名前を付けるフィールド又は式のテキスト
宛先フォルダ	FOLDER	[フォルダ] デフォルト：[一時フォルダに保存]	出力ファイルを保存するフォルダ。次のいずれかです： • 一時ディレクトリに保存 • ディレクトリに保存します

出力

ラベル	名前	タイプ	説明
フォルダ	FOLDER	[フォルダ]	出力ファイルを保存するフォルダ。

Python コード

Algorithm ID: native:extractbinary

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

フィールド計算機

フィールド計算機（:ref:`vector_expressions`を参照）を開きます。サポートされているすべての式と関数を使うことができます。

式の結果で新しいレイヤが作成されます。

フィールド計算機は **モデルデザイナー** で使用すると非常に便利です。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	計算するレイヤ
出力する属性（フィールド）の名前	FIELD_NAME	[文字列]	結果のフィールド名
フィールド型	FIELD_TYPE	[列挙型] デフォルト： 0	フィールドの型。次のいずれかです： • 0 --- Float • 1 --- 整数 • 2 --- 文字列 • 3 -- 日付
フィールド長	FIELD_LENGTH	[数値] デフォルト： 10	結果フィールドの長さ（最小 0）
フィールド精度	FIELD_PRECISION	[数値] デフォルト： 3	結果フィールドの精度（最小 0、最大 15）

次のページに続く

表 27.195 – 前のページからの続き

ラベル	名前	タイプ	説明
新しいフィールドを作る	NEW_FIELD	[ブール値] デフォルト： True	結果フィールドを新しいフィールドにする
計算式	FORMULA	[式]	結果を計算するために使う計算式
出力ファイル	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	出力レイヤの指定。 • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ベクタ：任意]	計算したフィールド値の出力レイヤ

Python コード

Algorithm ID: `native:fieldcalculator`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性をリファクタリング

ベクタレイヤの属性テーブルの構造を編集できます。

フィールドは、フィールドマッピングを使って、その型と名前を変更することができます。

元のレイヤは変更されません。提供されたフィールドマッピングに従って、変更された属性テーブルを含む新しいレイヤが生成されます。

注釈: フィールドに *constraints* を持つテンプレートレイヤーを使用する場合、その情報はウィジェットに

色付きの背景とツールチップで表示されます。この情報は設定時のヒントとして扱われます。制約が出力レイヤに追加されたり、アルゴリズムによってチェックされたり、強制されたりすることはありません。

リファクタ・フィールド・アルゴリズムは次が可能です：

- フィールド名と型を変更する
- フィールドを追加または削除する
- フィールドを並び替える
- 式を基づいて新しいフィールドを計算する
- 他のレイヤからフィールドリストを読み込む

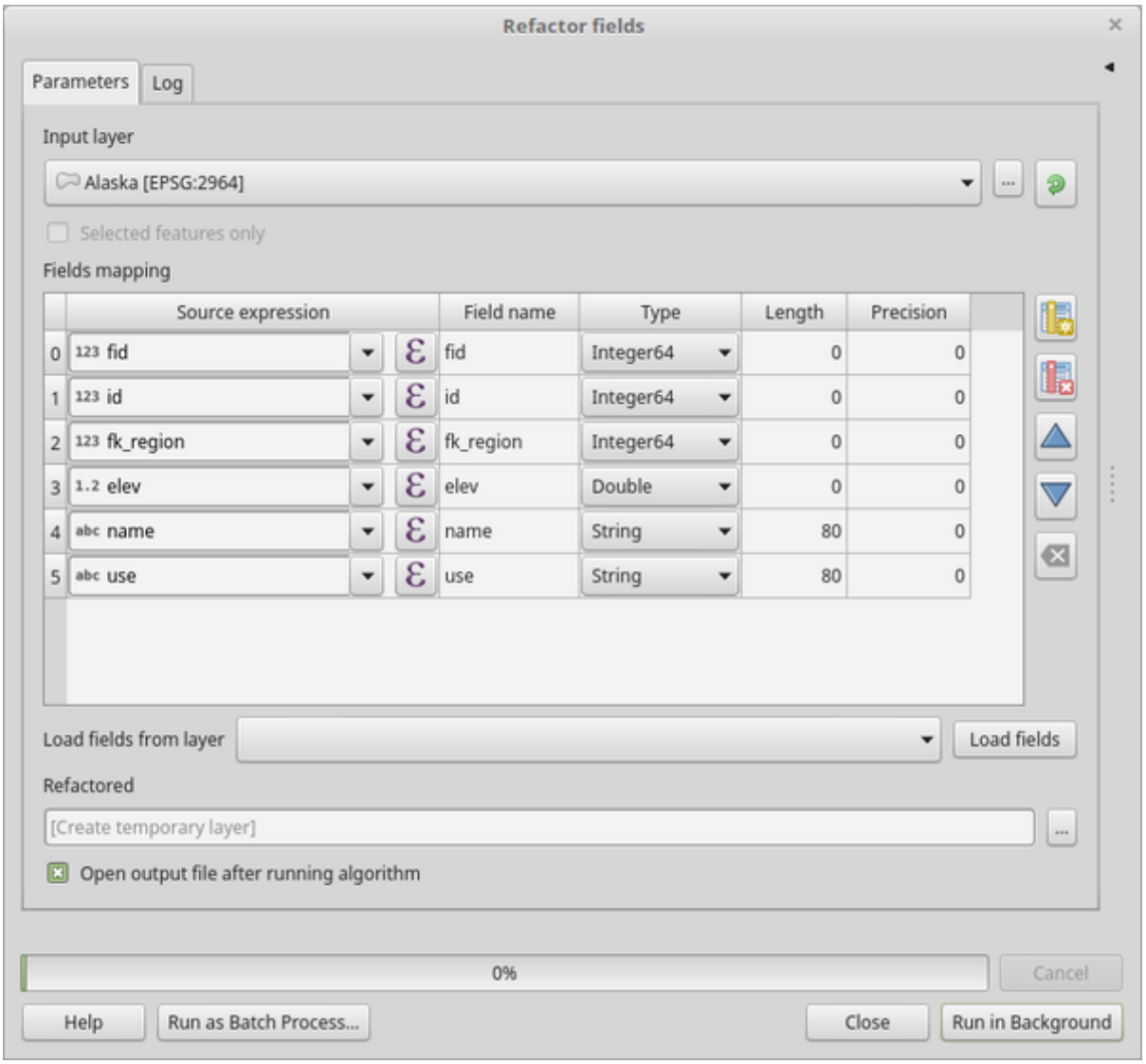


図 27.125: 属性をリファクタリングのダイアログ

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	変更するレイヤ

次のページに続く

表 27.196 – 前のページからの続き

ラベル	名前	タイプ	説明
属性の対応関係	FIELDS_MAPPING	[リスト]	出力フィールドとその定義の一覧表。埋め込みテーブルは、ソースレイヤーのすべてのフィールドを一覧表にし、それらを編集することができます： •  をクリックすると、新規にフィールドを作ります。 •  をクリックすると、フィールドが削除されます。 •  と  を使うと、選択したフィールドの順序が変わります。 •  をクリックすると、デフォルトのビューにリセットします。 再利用したいフィールドごとに、以下のオプションを入力する必要があります： **ソースの式 (式) [式]** 入力レイヤのフィールドまたは式 **フィールド名 (name) [文字列]** 出力レイヤのフィールドの名前。デフォルトでは入力フィールド名が維持されます。 **データ型 (type) [列挙]** 出力フィールドのデータ型。利用可能な型は、出力レイヤのプロバイダーによります。 **長さ (length) [数字]** 出力フィールドの長さ **精度 (precision) [数字]** 出力フィールドの精度 **制約 (constraints) [文字列]** テンプレートレイヤを使用している場合、テンプレートフィールドに制約が適用されているかどうかを示します。セルにカーソルを合わせると制約が表示されます。 **テンプレートレイヤからフィールドを読み込む** 現在のプロジェクトからレイヤをテンプレートとして選択し、(**読み込み** で) 上の「属性の対応関係」オプションにフィールドとその定義を入力することができます。

次のページに続く

表 27.196 – 前のページからの続き

ラベル	名前	タイプ	説明
再構成レイヤ	OUTPUT	[ベクタ：任意] デフォルト：[一時レイヤを作成]	出力レイヤの指定。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
再構成レイヤ	OUTPUT	[ベクタ：任意]	リファクタリングされた属性を持つ出力レイヤ

Python コード

Algorithm ID: native:refactorfields

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性名を変更

ベクタレイヤの既存のフィールドの名前を変更します。

元のレイヤは変更されません。属性テーブルが名前を変更されたフィールドを含んでいる新しいレイヤが生成されます。

参考:

属性をリファクタリング

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
変更する属性名	FIELD	[テーブルのフィールド：任意]	変更されるフィールド
新規属性名	NEW_NAME	[文字列]	新しいフィールド名
属性名変更済み出力	OUTPUT	[ベクタ: 入力と同じ] デフォルト：[一時レイヤを作成]	出力レイヤの指定。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
属性名変更済み出力	OUTPUT	[ベクタ: 入力と同じ]	名前を変更されたフィールドを持った出力レイヤ

Python コード

Algorithm ID: `qgis:renametablefield`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

属性を選択保持

ベクタレイヤをとり、選択されたフィールドのみを保持する新しいレイヤを生成します。他のフィールドはすべて削除されます。

参考:

属性を削除

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
保持するフィールド	FIELDS	[テーブルのフィールド：任意] [リスト]	レイヤの残しておくフィールドのリスト
保持されたフィールド	OUTPUT	[ベクタ: 入力と同じ] デフォルト：[一時レイヤを作成]	出力レイヤの指定。次のいずれかです: • 一時レイヤを作成（TEMPORARY_OUTPUT） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
保持されたフィールド	OUTPUT	[ベクタ: 入力と同じ]	保持されたフィールドを持った出力レイヤ

Python コード

Algorithm ID: native:retainfields

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズム**

をコンソールから使う を参照してください。

文字列を浮動小数点に変換

ベクタレイヤの指定された属性の型を変更し、数値文字列を含むテキスト属性を数値属性に変換します（例えば '1' を `1.0` に）。

アルゴリズムは新しいベクタレイヤを生成するので元のベクタレイヤは変更されません。

変換できない場合、選択した列は `NULL` 値を持つことになります。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	`INPUT`	[ベクタ：任意]	入力ベクタレイヤ
浮動小数点に変換する属性	`FIELD`	[テーブルのフィールド：文字列]	浮動小数点数フィールドに変換される入力レイヤの文字列フィールド。
出力レイヤ	`OUTPUT`	[入力レイヤと同じ] デフォルト：[一時レイヤを作成]	出力レイヤを指定します。次のいずれかです： • 一時レイヤを作成（`TEMPORARY_OUTPUT`） • ファイルに保存... • GeoPackage に保存... • データベーステーブルに保存... • レイヤに追加... ここでファイルの文字コードを変更することもできます。

出力

ラベル	名前	タイプ	説明
出力レイヤ	`OUTPUT`	[入力レイヤと同じ]	文字列フィールドを浮動小数点フィールドに変換した出力ベクタレイヤ

Python コード

Algorithm ID: `qgis:texttofloat`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.1.22 ベクタタイル

ベクタタイルを書き出し (WMBTiles)

1 つまたは複数のベクタレイヤを、高速なマップレンダリングと小さなデータサイズに最適化されたデータ形式であるベクタタイルにエクスポートします。

MBTiles は、タイル化された地図データを SQLite データベースに保存し、すぐに利用したり転送したりするための仕様です。MBTiles ファイルはタイルセットとして知られています。

パラメータ

ラベル	名前	データ型	説明
入力レイヤ	INPUT	[ベクタ：任意] [リスト]	ベクタタイルを生成するために結合するレイヤのリスト
最小ズーム値	MIN_ZOOM	[数値] デフォルト： 0	タイルセットがデータを提供する最も低いズームレベル。0 から 24 の間で設定します。
最大ズーム値	MAX_ZOOM	[数値] デフォルト： 3	タイルセットがデータを提供する最も高いズームレベル。0 から 24 の間で設定します。
領域 オプション	EXTENT	[範囲] デフォルト：未設定	レンダリングされた地図領域の最大範囲。範囲はすべてのズームレベルでカバーされる領域を定義する必要があります。
メタデータ: 名前 オプション	META_NAME	[文字列]	タイルセットの名前
メタデータ: 説明 オプション	META_DESCRIPTION	[文字列]	タイルセットの内容の説明
メタデータ: 出典 オプション	META_ATTRIBUTION	[文字列]	マップのデータソースおよび/またはスタイルを説明する文字列。
メタデータ: バージョン オプション	META_VERSION	[文字列]	タイルセットのバージョン。これは MBTiles の仕様ではなく、タイルセット自身のリビジョンを指します。
メタデータ: 形式 オプション	META_TYPE	[文字列]	タイルセットの形式。値は `overlay` または `baselayer` のいずれかです。
メタデータ: 中央 オプション	META_CENTER	[文字列]	地図のデフォルトビューの中心（コンマで区切られた数値の文字列：経度、緯度、ズームレベル）。例: `-122.1906,37.7599,11`
出力先タイル	OUTPUT	[ベクタタイル} デフォルト：[一時ファイルに保存]	出力 MBTiles ファイルを指定します。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	データ型	説明
出力先タイル	OUTPUT	[ファイル]	出力ベクタタイル .mbtiles ファイル。

Python コード

Algorithm ID: native:writevectortiles_mbtiles

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタタイルを書き出し (XYZ)

1 つまたは複数のベクタレイヤを、高速なマップレンダリングと小さなデータサイズに最適化されたデータ形式であるベクタタイルにエクスポートします。

パラメータ

ラベル	名前	データ型	説明
ファイルのテンプレート	XYZ_TEMPLATE	[文字列] デフォルト: '{z}/{x}/{y}.pbf'	ベクタタイル url を生成するためのテンプレート
入力レイヤ	INPUT	[ベクタ：任意] [リスト]	ベクタタイルを生成するために結合するレイヤのリスト
最小ズーム値	MIN_ZOOM	[数値] デフォルト： 0	タイルセットがデータを提供する最も低いズームレベル。0 から 24 の間で設定します。
最大ズーム値	MAX_ZOOM	[数値] デフォルト： 3	タイルセットがデータを提供する最も高いズームレベル。0 から 24 の間で設定します。
領域 オプション	EXTENT	[範囲] デフォルト：未設定	レンダリングされた地図領域の最大範囲。範囲はすべてのズームレベルでカバーされる領域を定義する必要があります。
出力フォルダ	OUTPUT_DIRECTORY	[フォルダ] デフォルト: [一時フォルダに保存]	出力ベクタタイルのフォルダを指定します。次のいずれかです: • 一時ディレクトリに保存 • ディレクトリに保存します

出力

ラベル	名前	データ型	説明
出力フォルダ	OUTPUT_DIRECTORY	[フォルダ]	ズームレベルに対応するサブフォルダに格納されたベクタタイルファイル（.pbf）の異なるサブセットを含むフォルダ。

Python コード

Algorithm ID: native:writevectortiles_xyz

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2 GDAL アルゴリズムプロバイダ

GDAL (Geospatial Data Abstraction Library) は、ラスタとベクタの地理空間データフォーマットのための翻訳ライブラリである。プロセシングフレームワークのアルゴリズムは GDAL ラスタープログラム と GDAL ベクタープログラム から派生しています。

27.2.1 ラスタ分析

傾斜方位

任意の GDAL サポートの標高ラスタから傾斜方位を作成します。傾斜方位は、斜面が向くコンパス方位です。ピクセルは、方位角を示す北からの度で測定された 0〜360 ° の値を持ちます。北半球では、斜面の北側は日陰になることが多く（方位角が 0 ° -90 ° と小さい）、南側は日射を多く受けます（方位角が 180 ° -270 ° と大きい）。

このアルゴリズムは GDAL DEM utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高として使用するバンドの番号
方位角の代わりに三角関数角を返す	TRIG_ANGLE	[ブール値] デフォルト： False	三角関数角をアクティブにすると、異なる分類になります：0 °（東）、90 °（北）、180 °（西）、270 °（南）。
平らな場合、-9999 ではなく 0 を返す	ZERO_FLAT	[ブール値] デフォルト： False	このオプションをアクティブにすると値が-9999 の平坦な区域に値 0 を挿入します。
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト： False	標高ラスタから境界を生成します
Horn の公式ではなく、Zevenbergen Thorne の公式を使用	ZEVENBERGEN	[ブール値] デフォルト： False	滑らかな地形に Zevenbergen Thorne の公式をアクティブにする

次のページに続く

表 27.199 – 前のページからの続き

ラベル	名前	タイプ	説明
傾斜方位（aspect）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤ。次のうちのひとつ: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
傾斜方位（aspect）	OUTPUT	[ラスタ]	角度の値を度で持った出力ラスタ

Python コード

Algorithm ID: `gdal:aspect`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

段彩図

任意の GDAL サポートの標高ラスタから段彩図マップを生成します。段彩図は特に標高を描写するために使用できます。アルゴリズムは、標高とテキストベースの色構成ファイルから計算された値を持つ 4 バンドラスタを出力します。デフォルトでは、与えられた標高値の間の色は滑らかにブレンドされ、その結果、きれいに色付けされた標高ラスタが得られます。

このアルゴリズムは GDAL DEM utility から派生したものです。

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高として使用するバンドの番号
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト： False	標高ラスタから境界を生成します
色構成ファイル	COLOR_TABLE	[ファイル]	テキスト型の色構成ファイル
マッチングモード	MATCH_MODE	[列挙型] デフォルト： 2	次のいずれかです： • 0 -- 厳密なカラーマッチングを使用する • 1 -- 最も近い RGBA を使う • 2 -- 滑らかにブレンドされた色を使用する
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： ''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します
段彩図	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤ。次のうちのひとつ: • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	タイプ	説明
段彩図	OUTPUT	[ラスタ]	4 バンド出力ラスタ

Python コード

アルゴリズム ID: gdal:colorrelief

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

nodata を埋める

nodata 値のラスタ領域を縁から内挿によって埋めます。nodata 領域の値は、逆距離加重を使い、周辺のピクセル値によって計算されます。内挿の後、結果の平滑化が行われます。入力は GDAL でサポートされている任意のラスタレイヤとすることができます。このアルゴリズムは、一般的に、（例えば標高モデルなどの）かなり連続的に変化するラスタの欠落領域を内挿するのに適しています。また、（航空写真のような）より不規則に変化する画像における小孔や亀裂を埋めるのにも適しています。一般的に、まばらなポイントデータからラスタを補間するにはあまり適していません。

このアルゴリズムは GDAL fillnodata ユーティリティ から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	操作するバンド。nodata 値は値 0 で表されていなければなりません。
内挿値を検索する距離（ピクセル単位）	DISTANCE	[数値] デフォルト： 10	内装する値を見つけるために全方向を検索するピクセルの数

次のページに続く

表 27.202 – 前のページからの続き

ラベル	名前	タイプ	説明
内挿後に実行するスムージング回数	ITERATIONS	[数値] デフォルト： 0	内装結果を平滑化するために実行する 3x3 フィルターパスの数（0 以上）。
デフォルトのマスクを使用しない	NO_MASK	[ブール値] デフォルト： False	ユーザー定義の有効マスクを使います
有効マスク	MASK_LAYER	[ラスタ]	埋める区域を決めるラスタレイヤ。
出力ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： "	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力ラスタ	OUTPUT	[ラスタ]	出力ラスタ

Python コード

Algorithm ID: `gdal:fillnodata`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッド（データメトリクス）

指定されたウィンドウと出力グリッドジオメトリを使用して、いくつかのデータメトリクスを計算します。

このアルゴリズムは、 GDAL grid utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

参考:

GDAL グリッドチュートリアル

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ（点）	`INPUT`	[ベクタ：ポイント]	入力ポイントベクタレイヤ

次のページに続く

表 27.204 – 前のページからの続き

ラベル	名前	タイプ	説明
使用するデータメトリック	METRIC	[列挙型] デフォルト： 0	次のいずれかです： • 0 -- 最小、グリッドノードの検索楕円で見つかった最小値 • 1 --- 最大、グリッドノードの検索楕円で見つかった最大値 • 2 --- 範囲、グリッドノードの検索楕円で見つかった最小値と最大値の差 • 3 --- カウント、グリッドノードの検索楕円で見つかったデータ点の数 • 4 --- 平均距離：グリッドノード（検索楕円の中心）とグリッドノードの検索楕円で見つかった全てのデータ点間の平均距離 • 5 --- 点間距離の平均、グリッドノードの検索楕円内で検出されたデータ点間の平均距離。楕円内の各点のペア間の距離が計算され、すべての距離の平均がグリッドノードの値として設定されます
検索楕円の第１半径	RADIUS_1	[数値] デフォルト： 0.0	検索楕円の最初の半径（回転角度が 0 の場合は X 軸）
検索楕円の第２半径	RADIUS_2	[数値] デフォルト： 0.0	探索楕円の二番目の半径（回転角度が 0 の場合は Y 軸）
** 検索楕円の回転角（反時計回り、単位は度）**	ANGLE	[数値] デフォルト： 0.0	楕円の回転角度（度）。楕円を反時計回りに回転させます。
採用するデータ数の下限	MIN_POINTS	[数値] デフォルト： 0.0	平均化する最小データ点数。検出された点数が少ない場合、グリッド・ノードは空とみなされ、NODATA マーカーで埋められます。
nodata 値	NODATA	[数値] デフォルト： 0.0	空の点を埋めるデータなしマーカー
内挿（データメトリクス）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
内挿する Z 値の属性 オプション	Z_FIELD	[テーブルのフィールド：数値]	内挿するフィールド
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します
出力のデータ型	DATA_TYPE	[列挙型] デフォルト：5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
内挿（データメトリクス）	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: gdal:griddatametrics

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッド（最近傍探索 IDW）

最近傍法にパワーグリッド法を組み合わせた逆距離を計算します。使用するデータ点の最大数が必要な場合に最適です。

このアルゴリズムは、 GDAL grid utility から派生したものです。

参考:

GDAL グリッドチュートリアル

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ（点）	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
重み付けの累乗	POWER	[数値] デフォルト： 2.0	重み付けの累乗
**スムージング係数*:	SMOOTHING	[数値] デフォルト： 0.0	スムージングパラメータ
検索円の半径	RADIUS	[数値] デフォルト： 1.0	検索円の半径
使用するデータ点の最大数	MAX_POINTS	[数値] デフォルト： 12	この数より多くの点を検索しません。

次のページに続く

表 27.206 – 前のページからの続き

ラベル	名前	タイプ	説明
採用するデータ数の下限	MIN_POINTS	[数値] デフォルト： 0	平均化する最小データ点数。検出された点数が少ない場合、グリッド・ノードは空とみなされ、NODATA マーカーで埋められます。
nodata 値	NODATA	[数値] デフォルト： 0.0	空の点を埋めるデータなしマーカー
内挿（最近傍探索による IDW）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
内挿する Z 値の属性 オプション	Z_FIELD	[テーブルのフィールド：数値]	内挿するフィールド
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： "	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

次のページに続く

表 27.207 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
内挿（最近傍探索による IDW）	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: `gdal:gridinversedistancenearestneighbor`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプ

ロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッド（累乗逆距離加重法）

累乗逆距離加重グリッド法は、加重平均補完器です。

すべてのデータポイントの座標と出力グリッドジオメトリを含む散布データ値と入力配列を与えてください。この関数は、出力グリッド内の指定された位置の補間値を計算します。

このアルゴリズムは、 GDAL grid utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

参考:

GDAL グリッドチュートリアル

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ（点）	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
重み付けの累乗	POWER	[数値] デフォルト： 2.0	重み付けの累乗
スムージング係数	SMOOTHING	[数値] デフォルト： 0.0	スムージングパラメータ
検索楕円の第１半径	RADIUS_1	[数値] デフォルト： 0.0	検索楕円の最初の半径（回転角度が 0 の場合は X 軸）
検索楕円の第２半径	RADIUS_2	[数値] デフォルト： 0.0	探索楕円の二番目の半径（回転角度が 0 の場合は Y 軸）
** 検索楕円の回転角（反時計回り、単位は度）**	ANGLE	[数値] デフォルト： 0.0	楕円の回転角度（度）。楕円を反時計回りに回転させます。
使用するデータ点の最大数	MAX_POINTS	[数値] デフォルト： 0	この数より多くの点を検索しません。
採用するデータ数の下限	MIN_POINTS	[数値] デフォルト： 0	平均化する最小データ点数。検出された点数が少ない場合、グリッド・ノードは空とみなされ、NODATA マーカーで埋められます。
nodata 値	NODATA	[数値] デフォルト： 0.0	空の点を埋めるデータなしマーカー

次のページに続く

表 27.208 – 前のページからの続き

ラベル	名前	タイプ	説明
IDW 内挿出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
内挿する Z 値の属性 オプション	Z_FIELD	[テーブルのフィールド：数値]	内挿するフィールド
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

次のページに続く

表 27.209 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
IDW 内挿出力	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: `gdal:gridinversedistance`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッド（線形）

Linear 法は、点群のドロネー三角形分割を計算し、その三角形分割のどの三角形に点があるかを調べ、その三角形内の重心座標から線形補間を行うことで線形補間を実行します。半径に応じて、点がどの三角形にもない場合、アルゴリズムは最も近い点の値または NODATA 値を使用します。

このアルゴリズムは、 GDAL grid utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ（点）	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
検索距離	RADIUS	[数値] デフォルト: -1.0	補間する点がドロネー三角形の三角形に収まらない場合、その最大距離を使って最近傍を探索するか、そうでなければ nodata を使います。もし -1 に設定すると、探索距離は無限大になります。もし 0 に設定すると、no data 値が使われます。
nodata 値	NODATA	[数値] デフォルト： 0.0	空の点を埋めるデータなしマーカー
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
内挿する Z 値の属性 オプション	Z_FIELD	[テーブルのフィールド：数値]	内挿するフィールド

次のページに続く

表 27.211 – 前のページからの続き

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します
出力のデータ型	DATA_TYPE	[列挙型] デフォルト：5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: gdal:gridlinear

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッド（移動平均）

移動平均は、単純なデータ平均化アルゴリズムです。楕円形の移動ウィンドウを使用して値を検索し、ウィンドウ内のすべてのデータポイントを平均します。検索楕円は指定された角度、グリッドノードにある楕円の中心で回転させることができます。ウィンドウ内に十分なポイントがない場合、グリッドノードは空と見なされ、指定された NODATA 値で埋められます。

このアルゴリズムは、 GDAL grid utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

参考:

GDAL グリッドチュートリアル

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ（点）	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
検索楕円の第１半径	RADIUS_1	[数値] デフォルト： 0.0	検索楕円の最初の半径（回転角度が 0 の場合は X 軸）

次のページに続く

表 27.212 – 前のページからの続き

ラベル	名前	タイプ	説明
検索楕円の第２半径	RADIUS_2	[数値] デフォルト： 0.0	探索楕円の二番目の半径（回転角度が 0 の場合は Y 軸）
** 検索楕円の回転角（反時計回り、単位は度）**	ANGLE	[数値] デフォルト： 0.0	楕円の回転角度（度）。楕円を反時計回りに回転させます。
採用するデータ数の下限	MIN_POINTS	[数値] デフォルト： 0.0	平均化する最小データ点数。検出された点数が少ない場合、グリッド・ノードは空とみなされ、NODATA マーカーで埋められます。
nodata 値	NODATA	[数値] デフォルト： 0.0	空の点を埋めるデータなしマーカー
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
内挿する Z 値の属性 オプション	Z_FIELD	[テーブルのフィールド：数値]	内挿するフィールド
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： ''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

次のページに続く

表 27.213 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: `gdal:gridaverage`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

グリッド（最近傍法）

最近傍法は補間または平滑化は何も行わず、グリッドノード検索楕円で見つかった最も近い点の値を取り、それを結果として返すだけです。何もポイントが見つからない場合は、指定された NODATA 値が返されます。

このアルゴリズムは、 GDAL grid utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

参考:

GDAL グリッドチュートリアル

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ（点）	INPUT	[ベクタ：ポイント]	入力ポイントベクタレイヤ
検索楕円の第１半径	RADIUS_1	[数値] デフォルト： 0.0	検索楕円の最初の半径（回転角度が 0 の場合は X 軸）
検索楕円の第２半径	RADIUS_2	[数値] デフォルト： 0.0	探索楕円の二番目の半径（回転角度が 0 の場合は Y 軸）
** 検索楕円の回転角（反時計回り、単位は度）**	ANGLE	[数値] デフォルト： 0.0	楕円の回転角度（度）。楕円を反時計回りに回転させます。
nodata 値	NODATA	[数値] デフォルト： 0.0	空の点を埋めるデータなしマーカー
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
内挿する Z 値の属性 オプション	Z_FIELD	[テーブルのフィールド：数値]	内挿するフィールド

次のページに続く

表 27.215 – 前のページからの続き

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： ''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: gdal:gridnearestneighbor

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

陰影図（hillshade）

優れた陰影起伏効果を持つラスタを出力します。これは、地形を可視化するのに非常に便利です。必要に応じて、垂直方向と水平方向の単位の違いを考慮するために光源、垂直誇張係数および縮尺係数の方位と高度を指定できます。

このアルゴリズムは GDAL DEM utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高の情報を含んでいるバンド
Z 係数（垂直方向の誇張）	Z_FACTOR	[数値] デフォルト： 1.0	この係数は出力する標高ラスタの高さを誇張します
スケール比（垂直方向に水平方向の単位の比率。）	SCALE	[数値] デフォルト： 1.0	水平方向の単位に対する垂直方向の単位の比率

次のページに続く

表 27.216 – 前のページからの続き

ラベル	名前	タイプ	説明
ライトの方位角	AZIMUTH	[数値] デフォルト: 315.0	標高ラスタを照らす光の方位を度単位で定義します。ラスタの上部から来る場合は 0、東から来る場合は 90 度です。
ライトの高さ	ALTITUDE	[数値] デフォルト： 45.0	光の高さを度単位で定義します。光が標高ラスタの上方から来る場合は 90、レーキングライトの場合は 0。
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト：False	標高ラスタから境界を生成します
Hom の公式ではなく、Zevenbergen&Thorne の公式を使用	ZEVENBERGEN	[ブール値] デフォルト：False	滑らかな地形に Zevenbergen Thorne の公式をアクティブにする
複合シェーディング	COMBINED	[ブール値] デフォルト：False	
多方向シェーディング	MULTIDIRECTIONAL	[ブール値] デフォルト：False	
陰影図（hillshade）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	内挿値の出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： "	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
陰影図 (hillshade)	OUTPUT	[ラスタ]	内挿値の出力ラスタ

Python コード

Algorithm ID: gdal:hillshade

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

純黒化 (near black)

ほぼ白黒の境界を黒に変換します。

このアルゴリズムでは、画像をスキャンして、黒または白の襟周りのカスタムの色に近いか、正確に黒、白または一つ以上であるすべてのピクセルを設定しようとします。これは、モザイク時にカラーピクセルが透明として扱われるように、損失が多い圧縮をされた航空写真を「修正」するためによく使用されます。

このアルゴリズムは GDAL nearblack utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
黒 (白) からの距離	NEAR	[数値] デフォルト： 15	ピクセル値が黒、白、またはカスタム色からどれだけ離れていても、黒、白、またはカスタム色に近いとみなせるかを選択します。
純黒化ではなく純白化	WHITE	[ブール値] デフォルト： False	黒に近いピクセルではなく、白に近い (255) ピクセルを検索します

次のページに続く

表 27.218 – 前のページからの続き

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	出力ラスタ

Python コード

Algorithm ID: `gdal:nearblack`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

特定値までの距離 (raster distance)

各ピクセルの中心からターゲットピクセルとして識別される最も近いピクセルの中心までの距離を示すラスタ近接地図を生成します。ターゲットピクセルとは、ラスタピクセル値がターゲットピクセル値の集合内にあるような、ソースラスタ内のピクセルです。

このアルゴリズムは GDAL proximity utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高の情報を含んでいるバンド
ターゲットとなるピクセルの値のリスト オプション	VALUES	[文字列] デフォルト： ''	ターゲットピクセルとみなされる、ソース画像内のターゲットピクセル値のリスト。指定しない場合は、0 でないすべてのピクセルがターゲットピクセルとみなされます。
距離の単位	UNITS	[列挙型] デフォルト： 1	生成される距離をピクセル座標にするか、ジオリファレンス座標にするかを指定します。以下のいずれかです： • 0 -- ジオリファレンス座標 • 1 -- ピクセル座標
最大距離（これ以上の距離は nodata になる）[オプション] オプション	MAX_DISTANCE	[数値] デフォルト： 0.0	生成される最大距離。この距離を超えるピクセルには nodata 値が使用されます。nodata 値が提供されない場合、出力バンドにその nodata 値を問い合わせます。出力バンドに nodata 値がない場合は、65535 が使用されます。距離は、**距離の単位** の値に従って解釈されます。
最大距離以内のピクセルに（距離ではなく）固定値を付与する［オプション］ オプション	REPLACE	[数値] デフォルト： 0.0	距離値の代わりに、ターゲットピクセルから最大距離より近いすべてのピクセル（ターゲットピクセルを含む）に適用する値を指定します。
nodata 値［オプション］ オプション	NODATA	[数値] デフォルト： 0.0	出力ラスタに使う nodata 値を指定します

次のページに続く

表 27.220 – 前のページからの続き

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

次のページに続く

表 27.221 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号あり 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	出力ラスタ

Python コード

Algorithm ID: `gdal:proximity`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

粗度 (roughness)

標高から計算された値を有するシングルバンドラスタを出力します。粗さは、表面の凹凸の度合いです。これは、中心画素とその周辺セルの最大セル間の差によって計算されます。粗度の決定は地形標高データの解析における役割を果たしています。それは河川の形態の計算、気候学と自然地理学一般に有用です。

このアルゴリズムは GDAL DEM utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高として使用するバンドの番号
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト： False	標高ラスタから境界を生成します
粗度 (roughness)	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： "	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。

出力

ラベル	名前	タイプ	説明
粗度 (roughness)	OUTPUT	[ラスタ]	単バンド出力の粗度ラスタ。値-9999 が nodata 値に使われます。

Python コード

Algorithm ID: gdal:roughness

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ふるい

閾値サイズ（ピクセル単位）より小さいラスターポリゴンを削除し、それらを最大隣接ポリゴンのピクセル値で置き換えます。ラスタ地図上に小さな領域が大量にある場合に便利です。

このアルゴリズムは GDAL sieve utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
閾値	THRESHOLD	[数値] デフォルト： 10	このサイズより小さいラスタポリゴンだけが取り除かれます
上下左右の 4 方向でなく、8 方向の連結関係をチェックする	EIGHT_CONNECTEDN	[ブール値] デフォルト： False	4 方向の連結関係の代わりに 8 方向の連結関係を使います
デフォルトのマスクを使用しない	NO_MASK	[ブール値] デフォルト： False	

次のページに続く

表 27.224 – 前のページからの続き

ラベル	名前	タイプ	説明
有効マスク オプション	MASK_LAYER	[ラスタ]	デフォルトの代わりに使う有効マスク
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	出力ラスタレイヤ

Python コード

Algorithm ID: gdal:sieve

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

傾斜

GDAL でサポートされている標高ラスタから傾斜図を作成します。傾きは水平に対する傾きの角度です。傾斜値を指定するには、度か傾きのパーセントか、希望する方を選択できます。

このアルゴリズムは GDAL DEM utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高の情報を含んでいるバンド
水平方向に対する垂直方向の単位の比率	SCALE	[数値] デフォルト： 1.0	水平方向の単位に対する垂直方向の単位の比率
傾斜の単位はパーセント（デフォルトは度）	AS_PERCENT	[ブール値] デフォルト： False	傾斜を度ではなくパーセントで表します
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト： False	標高ラスタから境界を生成します
Hom の公式ではなく、Zevenbergen&Thorne の公式を使用	ZEVENBERGEN	[ブール値] デフォルト： False	滑らかな地形に Zevenbergen Thorne の公式をアクティブにする
傾斜（slope）	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
傾斜（slope）	OUTPUT	[ラスタ]	出力ラスタ

Python コード

Algorithm ID: `gdal:slope`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

Terrain Ruggedness Index (TRI)

標高から計算された値を持つシングルバンドラスタを出力します。TRI（Terrain Ruggedness Index）は、中央画素とその周囲のセルとの間の平均差として定義される地形凹凸指数を表します

このアルゴリズムは GDAL DEM utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高として使用するバンドの番号
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト： False	標高ラスタから境界を生成します
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： ''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	出力凹凸ラスタ。nodata 値として-9999 が使用されます。

Python コード

アルゴリズム ID: `gdal:triterrainruggednessindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

地形位置指数 (TPI)

標高から計算された値を持つシングルバンドラスタを出力します。TPI は、地形位置指数を表し、中央ピクセルとその周囲のセルの平均との間の差として定義されます。

このアルゴリズムは GDAL DEM utility から派生したものです。

デフォルトメニュー: **ラスタ　解析**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高値として使うバンドの　番号
境界も計算	COMPUTE_EDGES	[ブール値] デフォルト： False	標高ラスタから境界を生成します
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	出力ラスタ。

Python コード

Algorithm ID: `gdal:tpitopographicpositionindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.2 ラスタ変換

gdal2xyz

ラスタデータを XYZ アスキーファイル形式に変換します。

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	変換するラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、変換したいバンドを選択してください
カンマ区切りの値を出力する	CSV	[ブール値] デフォルト：False	出力ファイルのタイプをカンマ区切り値 (csv) にするかどうかを設定します。
出力ファイル	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	タイプ	説明
出力ファイル	INPUT	[テーブル]	ラスタバンドからエクスポートされた値を含んだテーブルファイル。

Python コード

アルゴリズム ID: `gdal:gdal2xyz`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

PCT を RGB に変換

8 ビットのパレット画像を 24 ビットの RGB に変換します。入力ファイルから擬似カラーバンドを求めるフォーマットの RGB ファイルに変換します。

このアルゴリズムは GDAL pct2rgb utility から派生したものです。

デフォルトメニュー: **ラスタ　変換**

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力する 8 ビットのラスタ画像
バンド番号	BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、変換したいバンドを選択してください
RGBA ファイルを生成	RGBA	[ブール値] デフォルト：False	出力ファイルのタイプを RGBA にするかどうかを設定します。
PCT を RGB に変換	OUTPUT	[ファイル] デフォルト：[一時ファイルに保存]	出力ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	タイプ	説明
PCT を RGB に変換	OUTPUT	[ラスタ]	24 ビット RGB ラスタ画像

Python コード

アルゴリズム ID: `gdal:pcttorgb`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタをベクタ化 (polygonize)

共通のピクセル値を共有するラスター内のピクセルのすべての連結領域に対してベクトルポリゴンを生成します。各ポリゴンは、そのポリゴンのピクセル値を示す属性とともに生成されます。

このアルゴリズムは GDAL polygonize utility から派生したものです。

デフォルトメニュー: **ラスタ　変換**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト：入力レイヤの 1 番目のバンド	ラスタがマルチバンドの場合には、使いたいバンドを選択してください
作成する属性の名前	FIELD	[文字列] デフォルト：'DN'	連結領域の属性のフィールド名を指定します。
上下左右の 4 方向でなく、8 方向の連結関係をチェックする	EIGHT_CONNECTEDN	[ブール値] デフォルト： False	設定されていない場合、ラスターセルが連結されているとみなされるには、共通の境界線が必要です (*4-connected*)。設定されている場合、接触しているラスターセルも連結されているとみなされます (*8-connected*)。
ベクタ化	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時ファイルに保存]	出力する（ポリゴン）ベクタレイヤの指定。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
ベクタ化	OUTPUT	[ベクタ：ポリゴン]	出力ベクタレイヤ

Python コード

アルゴリズム ID: `gdal:polygonize`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

バンドを再構成

与えられたラスタレイヤから選択されたバンドを使って新しいラスタを作成します。このアルゴリズムではさらに新しく作成されたラスタでバンドの並べ替えを行うことができます。

このアルゴリズムは GDAL translate utility から派生したものです。

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
選択するバンド	BANDS	[ラスタバンド][リスト] デフォルト： なし	新しいラスタを作成するために使用するバンドの順序付きリスト
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： ''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 0	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- 入力レイヤのデータ型を使う • 1 --- Byte （8 ビット符号なし整数 (quint8)） • 2 --- Int16 （16 ビット符号あり整数 (qint16)） • 3 --- UInt16（16 ビット符号なし整数 (quint16)） • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Int32 （符号あり 32 ビット整数 (qint32)） • 6 -- Float32（32 ビット浮動小数点数 (float)） • 7 -- Float64（64 ビット浮動小数点数 (double)） • 8 -- CInt16 （複素数 Int16） • 9 -- CInt32 （複素数 Int32） • 10 -- CFloat32 （複素数 Float32） • 11 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ラスタ]	再構成されたバンドを持つ出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:rearrange_bands

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

RGB を PCT に変換

24 ビット RGB 画像を 8 ビットパレットに変換します。ダウンサンプリングされた RGB ヒストグラムに対して、メディアンカットアルゴリズムを用いて、与えられた RGB 画像に最適な擬似カラーテーブルを計算します．そして、そのカラーテーブルを用いて画像を擬似カラー画像に変換します。この変換は、出力画像の視覚的品質を最大化するために、フロイド-スタインバーグディザリング（誤差拡散）を利用します。

ラスターマップを分類したい場合であって階級の数を減らしたい場合は、このアルゴリズムで画像をダウンサンプリングするのが有効です。

このアルゴリズムは GDAL rgb2pct utility から派生したものです。

デフォルトメニュー: **ラスタ　変換**

パラメーター

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力 (RGB) ラスタレイヤ
色数	NCOLORS	[数値] デフォルト： 2	結果の画像に含まれる色の数。2〜256 の値が可能です。

次のページに続く

表 27.236 – 前のページからの続き

ラベル	名前	タイプ	説明
RGB を PCT に変換	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	タイプ	説明
RGB を PCT に変換	OUTPUT	[ラスタ]	出力ラスタレイヤ

Python コード

アルゴリズム ID: `gdal:rgbtopct`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

形式変換 (convert format)

異なる形式のラスタデータを変換します

このアルゴリズムは GDAL translate utility から派生したものです。

デフォルトメニュー: **ラスタ　　変換**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
出力ファイルの投影法を上書きする オプション	TARGET_CRS	[crs]	出力ファイルの投影法を指定します
指定した nodata 値を出力バンドに割り当てる オプション	NODATA	[数値] デフォルト：未設定	出力ラスタで nodata に使う値を定義します
各出力ファイルにこのファイルの全てのサブデータセットをコピーする	COPY_SUBDATASETS	[ブール値] デフォルト：False	サブデータセット用の個別ファイルを作成します
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力（変換された）ラスタレイヤの指定。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： "	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 0	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- 入力レイヤのデータ型を使う • 1 --- Byte （8 ビット符号なし整数 (quint8)） • 2 --- Int16 （16 ビット符号あり整数 (qint16)） • 3 --- UInt16（16 ビット符号なし整数 (quint16)） • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Int32 （符号あり 32 ビット整数 (qint32)） • 6 -- Float32（32 ビット浮動小数点数 (float)） • 7 -- Float64（64 ビット浮動小数点数 (double)） • 8 -- CInt16 （複素数 Int16） • 9 -- CInt32 （複素数 Int32） • 10 -- CFloat32 （複素数 Float32） • 11 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ラスタ]	出力（変換された）ラスタレイヤ

Python コード

アルゴリズム ID: gdal:translate

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.3 ラスタ抽出

範囲を指定して切り抜く

GDAL がサポートする任意のラスタファイルを、指定された範囲に切り抜きます。

このアルゴリズムは GDAL translate utility から派生したものです。

デフォルトメニュー: **ラスタ　抽出**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタ

次のページに続く

表 27.238 – 前のページからの続き

ラベル	名前	タイプ	説明
切り抜く範囲	EXTENT	[範囲]	出力ラスタに使用する範囲。指定したバウンディングボックス内のピクセルのみが出力に含まれます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
出力の CRS を上書きする	OVERCRS	[ブール値] デフォルト：False	チェックした場合、出力ファイルには入力レイヤの CRS が割り当てられます。
この nodata 値を出力バンドに割り当てる オプション	NODATA	[数値] デフォルト：なし	出力ラスタの nodata 値に挿入する値を定義する
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： ''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 0	出力ラスタファイルの形式を定義します。 オプション: • 0 -- 入力レイヤのデータ型を使う • 1 --- Byte （8 ビット符号なし整数 (quint8)） • 2 --- Int16 （16 ビット符号あり整数 (qint16)） • 3 --- UInt16（16 ビット符号なし整数 (quint16)） • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Int32 （符号あり 32 ビット整数 (qint32)） • 6 -- Float32（32 ビット浮動小数点数 (float)） • 7 -- Float64（64 ビット浮動小数点数 (double)） • 8 -- CInt16 （複素数 Int16） • 9 -- CInt32 （複素数 Int32） • 10 -- CFloat32 （複素数 Float32） • 11 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	指定された範囲で切り抜かれた出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:cliprasterbyextent

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

マスクレイヤで切り抜く

GDAL がサポートする任意のラスタをベクタマスクレイヤで切り抜きます。

このアルゴリズムは GDAL warp utility から派生したものです。

デフォルトメニュー: **ラスタ　抽出**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタ
マスクレイヤ	MASK	[ベクタ：ポリゴン]	ラスタを切り抜くためのベクタマスク
変換元 CRS	SOURCE_CRS	[crs]	入力ラスタに使う座標参照を設定します
変換先 CRS	TARGET_CRS	[crs]	マスクレイヤに使う座標参照を設定します

次のページに続く

表 27.239 – 前のページからの続き

ラベル	名前	タイプ	説明
変換先の領域 NEW in 3.24 オプション	TARGET_EXTENT	[範囲]	生成する出力ファイルの範囲 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • xmin, xmax, ymin, ymax として座標を入力
この nodata 値を出力バンドに割り当てる オプション	NODATA	[数値] デフォルト： なし	出力ラスタの nodata 値に挿入する値を定義する
アルファバンドを作る	ALPHA_BAND	[ブール値] デフォルト： False	結果のアルファバンドを作成します。アルファバンドには、ピクセルの透明度の値が含まれます。
マスクレイヤの領域に切り抜き範囲を一致させる	CROP_TO_CUTLINE	[ブール値] デフォルト： True	チェックするとベクタレイヤの範囲を出力ラスタに適用します。
入力ラスタの解像度を保持	KEEP_RESOLUTION	[ブール値] デフォルト： False	出力ラスタの解像度を変えません
出力ファイル解像度を設定	SET_RESOLUTION	[ブール値] デフォルト： False	出力解像度（セルサイズ）を指定するか
出力バンドの X 解像度 オプション	X_RESOLUTION	[数値] デフォルト： なし	出力ラスタのセルの幅
出力バンドの Y 解像度 オプション	Y_RESOLUTION	[数値] デフォルト： なし	出力ラスタのセルの高さ

次のページに続く

表 27.239 – 前のページからの続き

ラベル	名前	タイプ	説明
マルチスレッド実装を使用する	MULTITHREADING	[ブール値] デフォルト：False	2 つのスレッドが画像のチャンクを処理し、入出力操作を同時に実行するために使用されます。なお、計算自体はマルチスレッド化されません。
出力ファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト： "	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 0	出力ラスタファイルの形式を定義します。 オプション: • 0 -- 入力レイヤのデータ型を使う • 1 --- Byte （8 ビット符号なし整数 (quint8)） • 2 --- Int16 （16 ビット符号あり整数 (qint16)） • 3 --- UInt16（16 ビット符号なし整数 (quint16)） • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Int32 （符号あり 32 ビット整数 (qint32)） • 6 -- Float32（32 ビット浮動小数点数 (float)） • 7 -- Float64（64 ビット浮動小数点数 (double)） • 8 -- CInt16 （複素数 Int16） • 9 -- CInt32 （複素数 Int32） • 10 -- CFloat32 （複素数 Float32） • 11 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ラスタ]	ベクタレイヤによって切り抜かれた出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:cliprasterbymasklayer

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

等高線

GDAL がサポートしているあらゆる標高ラスタから等高線を抽出します

このアルゴリズムは GDAL contour utility から派生したものです。

デフォルトメニュー: **ラスタ　抽出**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	等高線を生成するラスタバンド
等高線の間隔	INTERVAL	[数値] デフォルト： 10.0	標高ラスタで指定されている単位による等高線の間隔を定義します（最小値 0）
属性名（空白なら標高は付加されません） オプション	FIELD_NAME	[文字列] デフォルト： 'ELEV'	標高を置く属性の名前を指定します。

次のページに続く

表 27.240 – 前のページからの続き

ラベル	名前	タイプ	説明
等高線の基準値 オプション	OFFSET	[数値] デフォルト： 0.0	
等高線	OUTPUT	[ベクタ：ライン] デフォルト：[一時ファイルに保存]	出力するベクタレイヤの指定。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
3D ベクタを生成	CREATE_3D	[ブール値] デフォルト： False	2D の代わりに 3D ベクタを強制的に作成する。各頂点の標高を含む。
全ラスタ値を有効とみなす	IGNORE_NODATA	[ブール値] デフォルト： False	データセットのあらゆる nodata 値を無視します。
nodata として扱うピクセル値 オプション	NODATA	[数値] デフォルト： なし	出力ラスタの nodata 値に挿入する値を定義する
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	追加の GDAL コマンドラインオプションを加えます。対応する GDAL ユーティリティのドキュメントを参照してください。

出力

ラベル	名前	タイプ	説明
等高線	OUTPUT	[ベクタ：ライン]	等高線の出力ベクタレイヤ

Python コード

アルゴリズム ID: `gdal:contour`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

等高線ポリゴン

GDAL がサポートしているあらゆる標高ラスタから等高線ポリゴンを抽出します

このアルゴリズムは GDAL contour utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	等高線を生成するラスタバンド
等高線の間隔	INTERVAL	[数値] デフォルト： 10.0	標高ラスタで指定されている単位による等高線の間隔を定義します（最小値 0）
等高線の基準値 オプション	OFFSET	[数値] デフォルト： 0.0	
ポリゴン最低標高の属性名 オプション	FIELD_NAME_MIN	[文字列] デフォルト：'ELEV_MIN'	等高線ポリゴンの最低標高を入れる属性名を指定します。指定されない場合、最低標高属性は付けられません。
ポリゴン最高標高の属性名 オプション	FIELD_NAME_MAX	[文字列] デフォルト：'ELEV_MAX'	等高線ポリゴンの最高標高を入れる属性名を指定します。指定されない場合、最高標高属性は付けられません。
等高線	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時ファイルに保存]	出力するベクタレイヤの指定。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
3D ベクタを生成	CREATE_3D	[ブール値] デフォルト：False	2D の代わりに 3D ベクタを強制的に作成する。各頂点の標高を含む。
全ラスタ値を有効とみなす	IGNORE_NODATA	[ブール値] デフォルト：False	データセットのあらゆる nodata 値を無視します。
nodata として扱うピクセル値 オプション	NODATA	[数値] デフォルト：なし	出力ラスタの nodata 値に挿入する値を定義する
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	追加の GDAL コマンドラインオプションを加えます。対応する GDAL ユーティリティのドキュメントを参照してください。

出力

ラベル	名前	タイプ	説明
等高線	OUTPUT	[ベクタ：ポリゴン]	等高線ポリゴンの出力ベクタレイヤ

Python コード

アルゴリズム ID: gdal:contour_polygon

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.4 ラスタその他

全体図を作成（ピラミッド）

ラスタレイヤのレンダリング時間を高速化するため、全体図（ピラミッド）を作成することができます。全体図は、QGIS がズームのレベルに応じて使用する、データの低解像度コピーです。

このアルゴリズムは GDAL addo utility から派生したものです。

デフォルトメニュー: **ラスタ　その他**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
すべの全体図を削除	CLEAN	[ブール値] デフォルト：False	ラスタから既存の全体図を削除します。 デフォルトは削除しません。

詳細パラメータ

ラベル	名前	タイプ	説明
全体図のレベル	LEVELS	[文字列] デフォルト: '2 4 8 16'	入力ラスターレイヤーの元の解像度から計算して全体図のレベルの数値を定義します。デフォルトでは、4 つのレベルが考慮されます。
リサンプリング方法 オプション	RESAMPLING	[列挙型] デフォルト： 0	定義されたリサンプリング法で全体図を計算します。可能なリサンプリング方法は以下の通りです： • 0 -- 最近傍 (nearest) • 1 -- 平均 (average) • 2 -- ガウス (gauss) • 3 -- キュービック畳み込み (cubic) • 4 -- B スプライン畳み込み (cubicspline) • 5 -- ランチョス窓関数 (lanczos) • 6 -- 平均 MP (average_mp) • 7 -- マグ/フェーズスペースの平均 (average_magphase) • 8 -- モード (mode)
全体図の形式 オプション	FORMAT	[列挙型] デフォルト： 0	全体図は、内部に保存することも、GTiff または ERDAS Imagine ファイルとして外部に保存することもできます。デフォルトでは、全体図は出力ラスタに保存されます。可能なフォーマット方法は以下の通りです： • 0 -- 内部（可能であれば） • 1 -- 外部 (GTiff .ovr) • 2 -- 外部 (ERDAS Imagine .aux)
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
Pyramidized	OUTPUT	[ラスタ]	全体図を持った出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:overviews

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

仮想ラスタを構築

入力 GDAL でサポートされているラスターのリストのモザイクである VRT（仮想データセット）を構築します。モザイクを使用すると、複数のラスターファイルをマージできます。

このアルゴリズムは GDAL buildvrt utility の派生です。

デフォルトメニュー: **ラスタ　その他**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	GDAL がサポートするラスタレイヤ。
Resolution	RESOLUTION	[列挙型] デフォルト： 0	モザイクの出力解像度。デフォルトでは、ラスタファイルの平均解像度が選択されます。 オプション: • 0 --- 平均 (average) • 1 --- Highest (highest) • 2 --- Lowest (lowest)

次のページに続く

表 27.243 – 前のページからの続き

ラベル	名前	タイプ	説明
Place each input file into a separate band	SEPARATE	[ブール値] デフォルト：False	'True' を指定すると、各ラスターファイルが VRT バンドで分離されたスタックバンドに入ることを定義できます。
投影法の違いを許す	PROJ_DIFFERENCE	[ブール値] デフォルト：False	出力バンドが、入力ラスターレイヤーの投影法から導き出された異なる投影法を持つことを許します。
仮想ラスタ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
ソースラスタに無い場合にアルファマスクバンドを VRT に追加する	ADD_ALPHA	[ブール値] デフォルト：False	ソースラスタに無い場合、アルファマスクバンドを VRT に追加します。
出力ファイルの投影法を上書きする オプション	ASSIGN_CRS	[crs] デフォルト：なし	出力ファイルの投影法を上書きします。再投影はされません。
Resampling algorithm	RESAMPLING	[列挙型] デフォルト：0	使用する再サンプリングアルゴリズム。次のいずれかです: • 0 --- Nearest Neighbour (`nearest`) • 1 --- Bilinear (`bilinear`) • 2 --- Cubic Convolution (`cubic`) • 3 --- B-Spline Convolution (`cubicspline`) • 4 --- Lanczos Windowed Sinc (`lanczos`) • 5 --- Average (`average`) • 6 --- Mode (`mode`)
Nodata value(s) for input bands (space separated) オプション	SRC_NODATA	[文字列] デフォルト：なし	空白で区切られた、入力バンドの Nodata 値
追加のコマンドラインパラメータ	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
仮想ラスタ	OUTPUT	[ラスタ]	出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:buildvirtualraster

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

gdal2tiles

OSGeo タイルマップサービス仕様書 に従った、小さなタイルとメタデータを含んだディレクトリを生成します。OpenGIS Web Map Tile Service Implementation Standard も参照してください。Google Maps、OpenLayers、Leaflet に基づいたビューアを持つシンプルなウェブページも生成されます。オンラインにある自分の地図をウェブブラウザで見るには、生成されたディレクトリをウェブサーバにアップロードするだけです。

提供された地図が EPSG:4326 投影法を使っている場合、このアルゴリズムは Google Earth (KML Super-Overlay) に必要なメタデータも生成します。

ESRI ワールドファイルと埋め込まれたジオリファレンスは、タイル生成時に使用されますが、適切なジオリファレンスなしで画像を公開することもできます。

このアルゴリズムは GDAL gdal2tiles utility から派生したものです。

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	GDAL がサポートするラスタレイヤ

次のページに続く

表 27.244 – 前のページからの続き

ラベル	名前	タイプ	説明
タイルカットプロファイル	PROFILE	[列挙型] デフォルト：0	次のいずれかです： • 0 -- メルカトル (mercator) • 1 -- 測地系 (geodetic) • 2 -- ラスタ (raster)
描画するズームレベル オプション	ZOOM	[文字列] デフォルト："	
生成する Web ビューア	VIEWER	[列挙型] デフォルト：0	次のいずれかです： • 0 -- すべて (all) • 1 -- GoogleMaps (google) • 2 -- OpenLayers (openlayers) • 3 -- Leaflet (leaflet) • 4 -- なし (none)
地図のタイトル オプション	TITLE	[文字列] デフォルト："	
地図の著作権	COPYRIGHT	[文字列] デフォルト："	
出力フォルダ	OUTPUT	[フォルダ] デフォルト：[一時フォルダに保存]	タイルの出力フォルダを指定します。次のいずれかです: • 一時ディレクトリに保存 • ディレクトリに保存します

詳細パラメータ

ラベル	名前	タイプ	説明
リサンプリング方法	RESAMPLING	[列挙型] デフォルト： 0	使用する再サンプリングアルゴリズム。次のいずれかです: • 0 --- 平均 (average) • 1 -- 最近傍 (near) • 2 -- バイリニア (bilinear) • 3 -- キュービック (cubic) • 4 -- キュービックスプライン (cubicspline) • 5 -- ランチョス窓関数 (lanczos) • 6 --- Antialias (antialias)
入力ソースデータの空間参照系 オプション	SOURCE_CRS	[crs] デフォルト： なし	
入力データに割り当てる透明度値 オプション	NODATA	[数値] デフォルト： 0.0	
生成されるタイルの URL オプション	URL	[文字列] デフォルト： ''	
Google Maps API キー (http://code.google.c オプション	GOOGLE_KEY	[文字列] デフォルト： ''	あなたの Google maps API キー
Bing Maps API キー (https://www.bingm オプション	BING_KEY	[文字列] デフォルト： ''	あなたの BING maps API キー
不足しているファイルのみを生成	RESUME	[ブール値] デフォルト： False	
Google Earth 用の KML を生成	KML	[ブール値] デフォルト： False	
EPSG:4326 の KML ファイルを自動生成しない	NO_KML	[ブール値] デフォルト： False	

出力

ラベル	名前	タイプ	説明
出力フォルダ	OUTPUT	[フォルダ]	出力フォルダ

Python コード

アルゴリズム ID: gdal:gdal2tiles

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

結合

簡単な方法でラスタファイルをマージします。ここでは、入力ラスタから擬似カラーテーブルを使用し、出力ラスタの種類を定義できます。すべての画像は同じ座標系でなければなりません。

このアルゴリズムは GDAL merge utility から派生したものです。

デフォルトメニュー: **ラスタ　その他**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ] [リスト]	入力ラスタレイヤ
最初のレイヤから疑似カラーテーブルを取得する	PCT	[ブール値] デフォルト： False	カラーリングに最初のレイヤの疑似カラーテーブルを使います
Place each input file into a separate band	SEPARATE	[ブール値] デフォルト： False	各ファイルを別のバンドに格納する

次のページに続く

表 27.245 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルの形式を定義します。オプションは以下のとおり: • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
nodata として扱うピクセル値 オプション	NODATA_INPUT	[数値] デフォルト：なし	このピクセル値を使って、マージされるファイルのピクセルを無視する
指定した nodata 値を出力バンドに割り当てる オプション	NODATA_OUTPUT	[数値] デフォルト：なし	指定した nodata 値を出力バンドに割り当てます。
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ラスタ]	出力ラスタレイヤ

Python コード

アルゴリズム ID: `gdal:merge`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

高解像度化

高解像度化処理を行います。（GeoTIFF など）「古典的な」出力データセットや高解像度処理を記述した VRT データセットを作成できます。

GDAL Pansharpen を参照。

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
低解像度のマルチバンドデータ	SPECTRAL	[ラスタ]	入力する（スペクトラル）ラスタレイヤ
高解像度の Panchromatic データ	PANCHROMATIC	[ラスタ]	入力する（パンクロマチック）ラスタレイヤ
出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力（高解像度化した）ラスタレイヤを指定します。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
Resampling algorithm	RESAMPLING	[列挙型] デフォルト：2	使用する再サンプリングアルゴリズム。次のいずれかです: • 0 --- Nearest Neighbour (nearest) • 1 --- Bilinear (bilinear) • 2 -- キュービック (cubic) • 3 -- キュービックスプライン (cubicspline) • 4 --- Lanczos Windowed Sinc (lanczos) • 5 --- Average (average)
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力	OUTPUT	[ラスタ]	出力（高解像度化した）ラスタレイヤ

Python コード

アルゴリズム ID: gdal:pansharp

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python

コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタ計算機

numpy 構文によるコマンドラインラスタ計算機。numpy の配列でサポートされている、+、-、 *、/などの基本的な算術演算と、>などの論理演算子を使います。すべての入力ラスタは同じ寸法でなければなりませんが、投影法のチェックは行われないことに注意してください。

GDAL ラスタ計算機ユーティリティのドキュメント を参照してください。

参考:

ラスタ計算機

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ A	INPUT_A	[ラスタ]	１番目の入力ラスタレイヤ（必須）
A のラスタバンド数	BAND_A	[ラスタのバンド]	入力レイヤ A のバンド（必須）
入力レイヤ B オプション	INPUT_B	[ラスタ] デフォルト： なし	２番目の入力ラスタレイヤ
B のラスタバンド数 オプション	BAND_B	[ラスタのバンド]	入力レイヤ B のバンド
入力レイヤ C オプション	INPUT_C	[ラスタ] デフォルト： なし	3 番目の入力ラスタレイヤ
C のラスタバンド数 オプション	BAND_C	[ラスタのバンド]	入力レイヤ C のバンド
入力レイヤ D オプション	INPUT_D	[ラスタ] デフォルト： なし	4 番目の入力ラスタレイヤ
D のラスタバンド数 オプション	BAND_D	[ラスタのバンド]	入力レイヤ D のバンド
入力レイヤ E オプション	INPUT_E	[ラスタ] デフォルト： なし	5 番目の入力ラスタレイヤ
E のラスタバンド数 オプション	BAND_E	[ラスタのバンド]	入力レイヤ E のバンド

次のページに続く

表 27.247 – 前のページからの続き

ラベル	名前	タイプ	説明
入力レイヤ F オプション	INPUT_F	[ラスタ]	6 番目の入力ラスタレイヤ
F のラスタバンド数 オプション	BAND_F	[ラスタのバンド] デフォルト：なし	入力レイヤ F のバンド
gdalnumeric 構文で計算。+-/*か numpy 配列関数 (logical_and()) を使用する	FORMULA	[文字列] デフォルト："	計算式。例: • A*(A>0) -- ラスタ A の値が 0 より大きければ、それを出力します。そうでなければ 0 を出力します。 • A*(A>0 and A>B) -- A の値が、0 より大きく且つ B の値より大きければ、それを出力します。そうでなければ 0 を出力します。 • A*logical_or(A<=177,A>=185) -- A <=177 又は A>185 のとき、A の値を出力します。そうでなければ 0 を出力します。 • sqrt(A*A+B*B) -- 二乗した A の値と二乗した B の値の和の平方根を出力します。
出力の nodata 値を設定 オプション	NO_DATA	[数値] デフォルト：なし	nodata に使う値

次のページに続く

表 27.247 – 前のページからの続き

ラベル	名前	タイプ	説明
出力領域 NEW in 3.24 オプション	INPUT	[範囲]	出力ラスタのカスタム範囲。GDAL 3.3+ でのみ利用可能。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
出力ラスタの型	RTYPE	[列挙型] デフォルト： 5	出力ラスタファイルのデータ型を定義します。オプションは以下のとおり： • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

次のページに続く

表 27.247 – 前のページからの続き

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	（計算した）出力ラスタレイヤを指定します。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト："	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ラスタ]	（計算した）出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:rastercalculator

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタの情報

gdalinfo プログラムは、GDAL がサポートするラスターデータセットに関する様々な情報を一覧表示します。

このアルゴリズムは GDAL info utility から派生したものです。

デフォルトメニュー: **ラスタ　その他**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力ラスタレイヤ
各バンドの最小・最大値を（メタ情報を使わず）実際に計算する	MIN_MAX	[ブール値] デフォルト：False	データセットの各バンドの最小・最大値を実際に計算する
画像統計量を読み込んで表示する（必要に応じて計算を強制する）	STATS	[ブール値] デフォルト：False	画像統計量を読み込んで表示する。画像に統計量が保存されていない場合は計算を強制します。
GCP 情報を出力しない	NO_GCP	[ブール値] デフォルト：False	地上基準点リストの印刷を抑制する。L1B AVHRR や HDF4 MODIS のように膨大な数の GCP を含むデータセットに便利です。
メタデータ情報を出力しない	NO_METADATA	[ブール値] デフォルト：False	メタデータの印刷を抑制する。データセットによっては多くのメタデータ文字列を含むことがあります。
出力ファイル	OUTPUT	[html] デフォルト：[一時ファイルに保存]	出力する HTML ファイルを指定します。次のいずれかです: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加のコマンドラインパラメータオプション	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[html]	入力ラスタレイヤの情報を含んでいる HTML ファイル

Python コード

アルゴリズム ID: gdal:gdalinfo

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

タイルを再タイル化

入力タイルのセットを再タイル化します。すべての入力タイルは、同じ座標系でジオリファレンスされており、一致したバンド数を持っていなければなりません。オプションでピラミッドレベルが生成されます。

このアルゴリズムは GDAL Retile utility から派生したものです。

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力ファイル	INPUT	[ラスタ] [リスト]	入力ラスタファイル
タイルの幅	TILE_SIZE_X	[数値] デフォルト： 256	ピクセルによるタイルの幅（最小 0）
タイルの高さ	TILE_SIZE_Y	[数値] デフォルト： 256	ピクセルによるタイルの高さ（最小 0）
隣接するタイルと重なるピクセル数	OVERLAP	[数値] デフォルト： 0	
ピラミッドを作成するレベル数	LEVELS	[数値] デフォルト： 1	最小値: 0

次のページに続く

表 27.251 – 前のページからの続き

ラベル	名前	タイプ	説明
出力フォルダ	OUTPUT	[フォルダ] デフォルト：[一時フォルダに保存]	タイルの出力フォルダを指定します。次のいずれかです: • 一時ディレクトリに保存 • ディレクトリに保存します
タイルのジオリファレンス情報を含む CSV ファイル	OUTPUT_CSV	[ファイル] デフォルト：[出力をスキップ]	タイルの出力ファイルを指定します。次のいずれかです: • 出力をスキップ • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
座標参照系 オプション	SOURCE_CRS	[crs] デフォルト：なし	
リサンプリング方法	RESAMPLING	[列挙型] デフォルト：0	使用する再サンプリングアルゴリズム。次のいずれかです: • 0 --- Nearest Neighbour (nearest) • 1 --- Bilinear (bilinear) • 2 -- キュービック (cubic) • 3 -- キュービックスプライン (cubicspline) • 4 --- Lanczos Windowed Sinc (lanczos)
CSV ファイルで使用されているカラム区切り文字 オプション	DELIMITER	[文字列] デフォルト: ';'	タイルのジオリファレンス情報を含んでいる CSV ファイルで使われている区切り文字
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。

次のページに続く

表 27.252 – 前のページからの続き

ラベル	名前	タイプ	説明
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：''	GDAL コマンドラインオプションを追加します
出力のデータ型	DATA_TYPE	[列挙型] デフォルト：5	出力ラスタファイルの形式を定義します。オプションは以下のとおり: • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）
ピラミッドのみ作成する	ONLY_PYRAMIDS	[ブール値] デフォルト：False	
タイルの各行 (row) に別々のディレクトリを使用する	DIR_FOR_ROW	[ブール値] デフォルト：False	

出力

ラベル	名前	タイプ	説明
出力フォルダ	OUTPUT	[フォルダ]	タイルの出力フォルダ。
タイルのジオリファレンス情報を含む CSV ファイル	OUTPUT_CSV	[ファイル]	タイルのジオリファレンス情報を持った CSV ファイル

Python コード

アルゴリズム ID: `gdal:retile`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

タイルインデックス

各入力ラスタファイル、ファイル名を含む属性、およびラスタの外形のポリゴンジオメトリのレコードを持つベクタレイヤを作成します。この出力は、ラスタタイルインデックスとして MapServer と共に使用するのに適しています。

このアルゴリズムは GDAL Tile Index utility から派生したものです。

デフォルトメニュー: **ラスタ　その他**

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力ファイル	`LAYERS`	[ラスタ] [リスト]	入力ラスタファイル。複数のファイルが可能です。
インデックス付きラスタのファイルパスを保持する属性の名前	`PATH_FIELD_NAME` Optional	[文字列] デフォルト: 'location'	インデックス付きラスタのファイルパスを保持する出力属性の名前。
絶対パスを保存する	`ABSOLUTE_PATH`	[ブール値] デフォルト： False	ラスタファイルへの絶対パスをタイルインデックスファイルに保存するかどうかを設定します。デフォルトでは、ラスタファイル名はコマンドで指定されたとおりにファイルに格納されます。
異なる CRS のファイルは無視する	`PROJ_DIFFERENCE`	[ブール値] デフォルト： False	タイルインデックスに既に挿入されているファイルと同じ投影法を持つファイルのみが挿入されます。デフォルトは投影法を確認せず、すべての入力を受け入れます。

次のページに続く

表 27.253 – 前のページからの続き

ラベル	名前	タイプ	説明
タイルインデックス	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時ファイルに保存]	インデックスを書き込むポリゴンベクタレイヤを指定します。以下のいずれか: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
指定 CRS でジオメトリを変換する オプション	TARGET_CRS	[crs]	入力ファイルのジオメトリは、指定した対象の座標参照系へ変換されます。デフォルトでは、入力ラスタと同じ座標参照系で単純な矩形のポリゴンが作成されます。
各タイルの CRS を格納するフィールドの名前 オプション	CRS_FIELD_NAME	[文字列]	各タイルの SRS を格納するフィールドの名前
CRS のフォーマット	CRS_FORMAT	[列挙型] デフォルト： 0	CRS のフォーマット。次のいずれかです: • 0 -- 自動 (AUTO) • 1 -- Well-known text (WKT) • 2 -- EPSG (EPSG) • 3 -- Proj.4 (PROJ)

出力

ラベル	名前	タイプ	説明
タイルインデックス	OUTPUT	[ベクタ：ポリゴン]	タイルインデックスを持ったポリゴンベクタレイヤ。

Python コード

アルゴリズム ID: `gdal:tileindex`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

見通し領域

ユーザー定義点に対する見通し領域ラスタを Wang2000 に定義されている方法で、入力ラスタ DEM から計算する。

パラメーター

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	入力標高ラスタレイヤ
バンド番号	BAND	[ラスタのバンド] デフォルト： 1	標高として使用するバンドの番号
視点の場所	OBSERVER	[ポイント]	視点の場所
視点の高さ	OBSERVER_HEIGHT	[数値] デフォルト： 1.0	DEM の単位による視点の高さ
ターゲットの高さ	TARGET_HEIGHT	[数値] デフォルト： 1.0	DEM の単位によるターゲット要素の高さ
視点からの最大距離	MAX_DISTANCE	[数値] デフォルト：100.0	DEM の単位による、視距離を計算する視点からの最大距離
出力	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤ。次のうちのひとつ: • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
追加のコマンドラインパラメータ	EXTRA	[文字列] デフォルト：なし	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
出力	OUTPUT	[ラスタ]	見通し領域を表示するラスタレイヤ

Python コード

アルゴリズム ID: `gdal:viewshed`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.5 ラスタ投影

投影法を設定

ラスタデータセットに座標系を適用します。

このアルゴリズムは、GDAL edit utility から派生したものです。

デフォルトメニュー: **ラスタ　投影法**

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT_LAYER	[ラスタ]	入力ラスタレイヤ
設定する座標参照系 (CRS)	CRS	[crs]	出力レイヤの投影法 (CRS)

出力

ラベル	名前	タイプ	説明
投影法を持ったレイヤ	OUTPUT	[ラスタ]	（新しい投影法の情報を持った）出力ラスタレイヤ

Python コード

アルゴリズム ID: gdal:assignprojection

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

投影法を抽出

ラスタファイルの投影法を抽出し、拡張子 .wld の *world* ファイルに書き込みます。

このアルゴリズムは GDAL srsinfo utility から派生したものです。

デフォルトメニュー: **ラスタ　投影法**

パラメータ

ラベル	名前	タイプ	説明
入力ファイル	INPUT_LAYER	[ラスタ]	入力ラスタ　アルゴリズムは生成される .wld ファイルの場所としてラスタファイルへのパスを使用するため、ラスタレイヤはファイル型でなければなりません。ファイル以外のラスタレイヤを使うとエラーになります。
.prj ファイルも作成する	PRJ_FILE_CREATE	[ブール値] デフォルト：False	これを有効にすると、投影法情報を含んだ .prj ファイルも作成されます。

出力

ラベル	名前	タイプ	説明
World file	WORLD_FILE	[ファイル]	ラスタファイルの変換パラメータを含んだ、拡張子 .wld のテキストファイル。
ESRI シェープファイル prj ファイル	PRJ_FILE	[ファイル]	CRS を記述する .prj 拡張子のテキストファイル。*.prj* **ファイルも作成する** が False の場合、None になります。

Python コード

アルゴリズム ID: gdal:extractprojection

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

再投影 (Warp)

ラスタレイヤを別の座標参照系 (CRS) に再投影します。出力ファイルの解像度とリサンプリング方法を選択できます。

このアルゴリズムは GDAL warp utility から派生したものです。

デフォルトメニュー: **ラスタ　投影法**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ラスタ]	再投影したい入力ラスタレイヤ
変換元 CRS オプション	SOURCE_CRS	[crs]	入力ラスタレイヤの CRS を定義します
ラスタの CRS オプション	TARGET_CRS	[crs] デフォルト：EPSG:4326	出力レイヤの CRS
リサンプリング法	RESAMPLING	[列挙型] デフォルト：0	使用するピクセル値のリサンプリング方法。オプション: • 0 -- 最近傍 • 1 -- バイリニア • 2 -- キュービック • 3 -- キュービックスプライン • 4 -- ランチョス窓関数 • 5 -- 平均 • 6 -- モード • 7 --- 最大 • 8 -- 最小値 • 9 -- 中央値 • 10 -- 第 1 四分位 (Q1) • 11 --- 第 3 四分位 (Q3)
出力バンドの no-data 値 オプション	NODATA	[数値] デフォルト：なし	出力バンドの nodata 値を設定します。指定しない場合、nodata 値はソースデータセットからコピーされます。
変換先 CRS の単位での解像度 オプション	TARGET_RESOLUTIO	[数値] デフォルト：なし	再投影結果の出力ファイルの解像度を定義します

次のページに続く

表 27.257 – 前のページからの続き

ラベル	名前	タイプ	説明
再投影したラスタファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。

次のページに続く

表 27.258 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 0	出力ラスタファイルの形式を定義します。オプションは以下のとおり: • 0 -- 入力レイヤのデータ型を使う • 1 --- Byte（8 ビット符号なし整数 (quint8)） • 2 --- Int16 （16 ビット符号あり整数 (qint16)） • 3 --- UInt16（16 ビット符号なし整数 (quint16)） • 4 --- UInt32（32 ビット符号なし整数 (quint32)） • 5 -- Int32 （符号あり 32 ビット整数 (qint32)） • 6 -- Float32（32 ビット浮動小数点数 (float)） • 7 -- Float64（64 ビット浮動小数点数 (double)） • 8 -- CInt16 （複素数 Int16） • 9 -- CInt32 （複素数 Int32） • 10 -- CFloat32 （複素数 Float32） • 11 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）

次のページに続く

表 27.258 – 前のページからの続き

ラベル	名前	タイプ	説明
出力ファイルの矩形範囲 オプション	TARGET_EXTENT	[範囲]	作成する出力ファイルのジオリファレンス範囲を設定します（デフォルトでは **変換先** *CRS* 内。指定された場合は **出力ファイルの矩形範囲の** *CRS* になります)。利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
出力ファイルの矩形範囲の CRS オプション	TARGET_EXTENT_CR	[crs]	出力ファイルの範囲に与えられた座標を解釈する CRS を指定します。これは出力データセットのターゲット CRS と混同しないでください。これは、例えば測地経度/緯度の CRS で出力座標を知っているが、投影座標系での結果が欲しいときなどの便宜のためです。
マルチスレッド実装を使用する	MULTITHREADING	[ブール値] デフォルト： False	2 つのスレッドが画像のチャンクを処理し、入出力操作を同時に実行するために使用されます。なお、計算自体はマルチスレッド化されません。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト： なし	追加の GDAL コマンドラインオプションを加えます。

出力

ラベル	名前	タイプ	説明
再投影したラスタファイル	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	再投影された出力ラスタレイヤ

Python コード

アルゴリズム ID: `gdal:warpreproject`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.6 ベクタ変換

フォーマット変換

OGR でサポートされているベクタレイヤを OGR でサポートされている別の形式に変換します。

このアルゴリズムは ogr2ogr utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
データセットの全レイヤを変換 NEW in 3.24	CONVERT_ALL_LAYE	[ブール値] デフォルト：False	データセット全体を変換します。このオプションでサポートされている出力形式は GPKG と GML です。
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力するベクタレイヤの指定。次のいずれかです: • 一時ファイルに保存 • ファイルに保存... ファイルに保存 では、出力形式を指定する必要があります。すべての GDAL ベクタ形式に対応しています。一時ファイルに保存 では、QGIS のデフォルトベクタ形式が使われます。

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[入力レイヤと同じ]	出力ベクタレイヤ

Python コード

Algorithm ID: `gdal:convertformat`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタ化（属性で上書き）

ラスタレイヤをベクタレイヤの値で上書きします。新しい値は、重なるベクトル地物の属性値に基づいて割り当てられます。

このアルゴリズムは GDAL rasterize utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	`INPUT`	[ベクタ：任意]	入力ベクタレイヤ
入力ラスタレイヤ	`INPUT_RASTER`	[ラスタ]	入力ラスタレイヤ
焼き込む値の属性 オプション	`FIELD`	[テーブルのフィールド：数値]	ピクセル値の設定に使用する属性フィールドを定義します

詳細パラメータ

ラベル	名前	タイプ	説明
ラスタの既存値に焼き込み値を加算	`ADD`	[ブール値] デフォルト：False	False の場合、ピクセルは選択されたフィールドの値が割り当てられます。True の場合、選択されたフィールドの値が入力ラスタレイヤの値に追加されます。
追加のコマンドラインパラメータ オプション	`EXTRA`	[文字列] デフォルト：''	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
Rasterized	OUTPUT	[ラスタ]	上書きされる入力ラスタレイヤ

Python コード

Algorithm ID: gdal:rasterize_over

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ラスタ化（固定値で上書き）

ラスタレイヤの一部を固定値で上書きします。上書きするピクセルは、提供された（重なる）ベクタレイヤを基に選択されます。

このアルゴリズムは GDAL rasterize utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
入力ラスタレイヤ	INPUT_RASTER	[ラスタ]	入力ラスタレイヤ
固定焼き込み値	BURN	[数値] デフォルト： 0.0	焼き込み値

詳細パラメータ

ラベル	名前	タイプ	説明
ラスタの既存値に焼き込み値を加算	ADD	[ブール値] デフォルト：False	False の場合、ピクセルには固定値が割り当てられます。True の場合、固定値は入力ラスタレイヤの値に追加されます。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト：''	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
Rasterized	OUTPUT	[ラスタ]	上書きされる入力ラスタレイヤ

Python コード

Algorithm ID: `gdal:rasterize_over_fixed_value`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタをラスタ化

ベクトルジオメトリ（点、ライン、ポリゴン）をラスタ画像に変換します。

このアルゴリズムは GDAL rasterize utility から派生したものです。

デフォルトメニュー: **ラスタ　変換**

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
焼き込む値の属性 オプション	FIELD	[テーブルのフィールド：数値]	ピクセルの属性を選択する属性フィールドを定義します
固定焼き込み値 オプション	BURN	[数値] デフォルト： 0.0	すべての地物について、バンドに書き込む固定値。
地物の Z 値をラスタ値に焼き込む オプション	USE_Z	[ブール値] デフォルト： False	地物の"Z"値から焼き込み値を抽出することを示します。点とライン（各セグメントに沿った線形補間）で機能します。ポリゴンの場合は、平らな場合（すべての頂点で同じ Z 値）にのみ正しく機能します。
出力ラスタサイズの単位	UNITS	[列挙型] デフォルト： 0	出力ラスタのサイズ／解像度を定義するときに使用する単位。次のいずれかです: • 0 -- ピクセル • 1 -- 地理単位
水平方向の解像度	WIDTH	[数値] デフォルト： 0.0	出力ラスタの幅（サイズの単位が「ピクセル」の場合）または水平解像度（サイズの単位が「地理単位」の場合）を設定します。最小値：0.0。
鉛直方向の解像度	HEIGHT	[数値] デフォルト： 0.0	出力ラスタの高さ（サイズの単位が「ピクセル」の場合）または垂直解像度（サイズの単位が「地理単位」の場合）を設定します。

次のページに続く

表 27.261 – 前のページからの続き

ラベル	名前	タイプ	説明
出力領域 オプション	EXTENT	[範囲]	出力ラスタ レイヤの範囲。範囲が指定されていない場合は、選択された参照レイヤをカバーする最小範囲が使用されます。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
指定した nodata 値を出力バンドに割り当てる オプション	NODATA	[数値] デフォルト： 0.0	指定した nodata 値を出力バンドに割り当てます
Rasterized	OUTPUT	[ラスタ] デフォルト：[一時ファイルに保存]	出力ラスタレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存... ファイルに保存 では、出力形式を指定する必要があります。すべての GDAL ラスタ形式に対応しています。一時ファイルに保存 では、QGIS のデフォルトラスタ形式が使われます。

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト："	作成するラスタを制御する 1 つ以上の作成オプションを追加します（色、ブロックサイズ、ファイル圧縮...）。便利なことに、定義済みのプロファイルを利用することができます (*GDAL* **ドライバのオプションセクション** を参照)。 バッチプロセスとモデルデザイナー: 複数のオプションをパイプ文字 (|) で区切ります。
出力のデータ型	DATA_TYPE	[列挙型] デフォルト： 5	出力ラスタファイルの形式を定義します。 オプション: • 0 -- Byte （符号なし 8 ビット整数 (quint8)） • 1 -- Int16 （符号付き 16 ビット整数 (qint16)） • 2 -- UInt16（符号なし 16 ビット整数 (quint16)） • 3 -- UInt32（符号なし 32 ビット整数 (quint32)） • 4 -- Int32 （符号あり 32 ビット整数 (qint32)） • 5 -- Float32（32 ビット浮動小数点数 (float)） • 6 -- Float64（64 ビット浮動小数点数 (double)） • 7 -- CInt16 （複素数 Int16） • 8 -- CInt32 （複素数 Int32） • 9 -- CFloat32 （複素数 Float32） • 10 -- CFloat64 （複素数 Float64） 利用可能なオプションは、QGIS と一緒にビルドされた GDAL のバージョンによって異なります（ **ヘルプ** *QGIS* **について** メニューを参照）
指定値で事前に初期化する オプション	INIT	[数値]	この値で出力画像バンドを事前に初期化します。出力ファイルの nodata 値としてマークされません。すべてのバンドで同じ値が使用されます。
逆ラスタ化	INVERT	[ブール値] デフォルト： False	固定焼き込み値、または最初の地物に関連付けられた焼き込み値を、指定されたポリゴンの内側でない画像のすべての部分に焼き付けます。
追加のコマンドラインパラメータ オプション	EXTRA	[文字列] デフォルト："	GDAL コマンドラインオプションを追加します

出力

ラベル	名前	タイプ	説明
Rasterized	OUTPUT	[ラスタ]	出力ラスタレイヤ

Python コード

Algorithm ID: gdal:rasterize

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示される ID です。 *parameter dictionary* は、パラメータの「名前」とその値を指定するマッピング型です。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.7 ベクタジオプロセシング

バッファを作成

ベクタレイヤの地物の周囲にバッファを作る

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
ジオメトリのカラム名	GEOMETRY	[文字列] デフォルト: 'geometry'	使用する入力レイヤのジオメトリ列の名前
バッファ距離	DISTANCE	[数値] デフォルト： 10.0	最小値: 0.0
属性でディゾルブする オプション	FIELD	[テーブルのフィールド：任意] デフォルト： なし	ディゾルブに使うフィールド

次のページに続く

表 27.262 – 前のページからの続き

ラベル	名前	タイプ	説明
結果をディゾルブする	DISSOLVE	[ブール値] デフォルト: False	**設定した場合、結果はディゾルブされます** ディゾルブするフィールドが設定されないときは、すべてのバッファがひとつの地物にディゾルブされます
各ジオメトリに対して 1 つの地物を生成する（シングルパート化）	EXPLODE_COLLECTI	[ブール値] デフォルト: False	
バッファ (buffer)	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時ファイルに保存]	バッファの出力レイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
バッファ (buffer)	OUTPUT	[ベクタ：ポリゴン]	出力ベクタレイヤ

Python コード

アルゴリズム ID: `gdal:buffervectors`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

矩形領域で切り抜く

OGR がサポートする任意のベクタファイルを、指定された範囲に切り取ります。

このアルゴリズムは GDAL ogr2ogr utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ

次のページに続く

表 27.264 – 前のページからの続き

ラベル	名前	タイプ	説明
切り抜く範囲	EXTENT	[範囲]	出力ベクタファイルに使用するバウンディングボックスを定義します。それはターゲット CRS 座標で定義されなければなりません。 利用可能な方法: • レイヤから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使います • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します • 現在のキャンバス領域を利用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします • `xmin, xmax, ymin, ymax` として座標を入力します
出力ファイル	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	（切り抜いた）出力レイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[入力レイヤと同じ]	（切り抜いた）出力レイヤ。デフォルト形式は「ESRI シェープファイル」。

Python コード

アルゴリズム ID: `gdal:clipvectorbyextent`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

マスクレイヤで切り抜く

OGR がサポートする任意のベクタレイヤを、マスクポリゴンレイヤで切り取ります。

このアルゴリズムは GDAL ogr2ogr utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
マスクレイヤ	MASK	[ベクタ：ポリゴン]	入力ベクタレイヤから切り抜く範囲として使うレイヤ
出力ファイル	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	（マスクされた）出力レイヤ。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[入力レイヤと同じ]	（マスクされた）出力レイヤ。デフォルト形式は「ESRI シェープファイル」。

Python コード

アルゴリズム ID: `gdal:clipvectorbypolygon`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

融合 (dissolve)

指定された属性/フィールドに同じ値を持つジオメトリを融合（結合）します。出力されるジオメトリはマルチパートです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	融合する入力レイヤ

次のページに続く

表 27.265 – 前のページからの続き

ラベル	名前	タイプ	説明
ディゾルブする属性 オプション	FIELD	[テーブルのフィールド：任意]	ディゾルブに使う入力レイヤの属性
ジオメトリのカラム名	GEOMETRY	[文字列] デフォルト: 'geometry'	ディゾルブに使う入力レイヤのジオメトリ列の名前
融合ポリゴンの出力	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力レイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
各ジオメトリに対して 1 つの地物を生成する（シングルパート化）	EXPLODE_COLLECTI	[ブール値] デフォルト: False	ソースファイルにあるどの種類のジオメトリコレクションであっても各ジオメトリに対して 1 つの地物を生成する（シングルパート化）
入力の属性を引き継ぐ	KEEP_ATTRIBUTES	[ブール値] デフォルト: False	入力レイヤのすべての属性を引き継ぎます
ディゾルブした地物の数	COUNT_FEATURES	[ブール値] デフォルト: False	ディゾルブした地物を数えて出力レイヤに含めます。
ディゾルブした地物の面積と周長を計算する	COMPUTE_AREA	[ブール値] デフォルト: False	**ディゾルブした地物の面積と周長を計算し、** それを出力レイヤに含めます
属性の最小／最大／合計／平均を計算する	COMPUTE_STATISTI	[ブール値] デフォルト: False	指定された数値属性の統計量（最小、最大、合計、平均）を計算し、出力レイヤに含めます
統計量を計算する属性 オプション	STATISTICS_ATTRI	[テーブルのフィールド：数値]	統計量を計算する数値属性
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
融合ポリゴンの出力	OUTPUT	[入力レイヤと同じ]	**出力マルチパートジオメトリレイヤ**（ディゾルブされたジオメトリを持った）

Python コード

アルゴリズム ID: `gdal:dissolve`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

曲線をオフセット

ラインを指定した距離だけオフセットします。正の値の距離はラインを左に、負の値の距離は右にオフセットします。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインレイヤ
ジオメトリのカラム名	GEOMETRY	[文字列] デフォルト： 'geometry'	使用する入力レイヤのジオメトリ列の名前
オフセット距離（左: 正、右: 負）	DISTANCE	[数値] デフォルト： 10.0	

次のページに続く

表 27.267 – 前のページからの続き

ラベル	名前	タイプ	説明
曲線をオフセット	OUTPUT	[ベクタ：ライン] デフォルト：[一時ファイルに保存]	出力ラインレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
曲線をオフセット	OUTPUT	[ベクタ：ライン]	出力オフセット曲線レイヤ

Python コード

アルゴリズム ID: `gdal:offsetcurve`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

片側バッファを作成

ラインベクタレイヤのラインの片側（右または左）にバッファを作ります。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインレイヤ
ジオメトリのカラム名	GEOMETRY	[文字列] デフォルト: 'geometry'	使用する入力レイヤのジオメトリ列の名前
バッファ距離	DISTANCE	[数値] デフォルト： 10.0	
バッファを作る側	BUFFER_SIDE	[列挙型] デフォルト：0	次のいずれかです： • 0 -- 右 • 1 -- 左
属性でディゾルブする オプション	FIELD	[テーブルのフィールド：任意] デフォルト： なし	ディゾルブに使うフィールド
結果をディゾルブする	DISSOLVE	[ブール値] デフォルト: False	**設定した場合、結果はディゾルブされます** ディゾルブするフィールドが設定されないときは、すべてのバッファがひとつの地物にディゾルブされます
各ジオメトリに対して 1 つの地物を生成する（シングルパート化）	EXPLODE_COLLECTI	[ブール値] デフォルト: False	
出力ファイル	OUTPUT	[ベクタ：ポリゴン] デフォルト：[一時ファイルに保存]	バッファの出力レイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[ベクタ：ポリゴン]	出力ベクタレイヤ

Python コード

アルゴリズム ID: `gdal:onesidebuffer`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

線に沿った点

ラインベクタレイヤの各線上に、始点からある距離で、点を生成する。距離は線の長さに対する比率で指定します。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：ライン]	入力ラインレイヤ

次のページに続く

表 27.269 – 前のページからの続き

ラベル	名前	タイプ	説明
ジオメトリのカラム名	GEOMETRY	[文字列] デフォルト: 'geometry'	使用する入力レイヤのジオメトリ列の名前
線の始点からの距離（全長との比率）	DISTANCE	[数値] デフォルト:0.5(線の中央)	
出力レイヤ	OUTPUT	[ベクタ：ポイント] デフォルト：[一時ファイルに保存]	出力ポイントレイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

詳細パラメータ

ラベル	名前	タイプ	説明
追加オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
出力レイヤ	OUTPUT	[ベクタ：ポイント]	出力ポイントレイヤ

Python コード

アルゴリズム ID: gdal:pointsalonglines

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.2.8 ベクタその他

仮想レイヤを構築

ベクタレイヤのセットを含んだ仮想ベクタレイヤを生成します。出力仮想ベクタレイヤは、現在のプロジェクトでは開かれません。

このアルゴリズムは、複数のレイヤを必要とするアルゴリズムが、レイヤが指定されている vrt を 1 つしか受け取らない場合に特に有用です。

パラメータ

ラベル	名前	タイプ	説明
入力データソース	INPUT	[ベクタ：任意] [リスト]	仮想ベクタを構築したいベクタレイヤを選択します
和集合の VRT を作成	UNIONED	[ブール値] デフォルト：False	すべてのベクタをひとつの vrt ファイルにまとめたいときにチェックします
Virtual vector	OUTPUT	[入力レイヤと同じ] デフォルト：[一時ファイルに保存]	出力レイヤを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存...

出力

ラベル	名前	タイプ	説明
Virtual vector	OUTPUT	[ベクタ：任意]	選んだソースから作られた出力仮想ベクタ

Python コード

アルゴリズム ID: gdal:buildvirtualvector

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

SQL を実行

ソースレイヤに対して、SQL 構文を用いた単純または複雑なクエリを実行します。クエリの結果は新しいレイヤとして追加されます。

このアルゴリズムは GDAL ogr2ogr utility から派生したものです。

パラメータ

基本パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	`INPUT`	[ベクタ：任意]	OGR がサポートする入力ベクタレイヤ
SQL 文	`SQL`	[文字列]	例えば、 `SELECT * FROM my_table WHERE name is not null` のように SQL クエリを定義します。
SQL ダイアレクト	`DIALECT`	[列挙型] デフォルト：0	使用する SQL ダイアレクト。次のいずれかです： • 0 -- なし • 1 -- OGR SQL • 2 -- SQLite
SQL の結果	`OUTPUT`	[ベクタ：任意]	出力レイヤの指定。次のいずれかです: • 一時ファイルに保存 • ファイルに保存... ファイルに保存 では、出力形式を指定する必要があります。すべての GDAL ベクタ形式に対応しています。一時ファイルに保存 の場合、デフォルトの出力ベクタレイヤ形式が使用されます。

詳細パラメータ

ラベル	名前	タイプ	説明
追加の生成オプション オプション	`OPTIONS`	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

ラベル	名前	タイプ	説明
SQL の結果	OUTPUT	[ベクタ：任意]	クエリによって作成されるベクタレイヤ

Python コード

アルゴリズム ID: gdal:executesql

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

PostgreSQL へ出力（既存接続）

利用可能な接続に基づいて PostgreSQL データベース内のベクタレイヤをインポートします。接続はあらかじめ **適切に定義されている** 必要があります。「ユーザー名を保存」と「パスワードを保存」のチェックボックスが有効になっていることに注意してください。その後、アルゴリズムを使用することができます。

このアルゴリズムは GDAL ogr2ogr utility から派生したものです。

パラメータ

ラベル	名前	タイプ	説明
データベース（接続名）	DATABASE	[文字列]	接続する PostgreSQL データベース
入力レイヤ	INPUT	[ベクタ：任意]	データベースに出力する OGR がサポートしているベクタレイヤ
シェープエンコーディング オプション	SHAPE_ENCODING	[文字列] デフォルト："	データに適用するエンコーディングを設定します

次のページに続く

表 27.271 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のジオメトリ型	GTYPE	[列挙型] デフォルト： 0	出力のジオメトリ型を定義します。次のいずれかです： • 0 --- • 1 --- NONE • 2 --- GEOMETRY • 3 --- POINT • 4 --- LINESTRING • 5 --- POLYGON • 6 --- GEOMETRYCOLLECTION • 7 --- MULTIPOINT • 8 --- MULTIPOLYGON • 9 --- MULTILINESTRING
CRS を出力に割り当てる オプション	A_SRS	[crs] デフォルト： なし	データベーステーブルの出力 CRS を定義します
出力を再投影する CRS オプション	T_SRS	[crs] デフォルト： なし	出力するときにこの CRS に再投影／変換します
CRS を上書きする オプション	S_SRS	[crs] デフォルト： なし	入力レイヤ CRS を上書きします
スキーマ（スキーマ名） オプション	SCHEMA	[文字列] デフォルト： 'public'	データベーステーブルのスキーマを定義します
エクスポートするテーブル（空白の場合、レイヤ名が使われます） オプション	TABLE	[文字列] デフォルト： ''	データベースにインポートされるテーブルの名前を定義します。テーブルの名前のデフォルトは、入力ベクタファイルの名前です。
主キー（新規フィールド） オプション	PK	[文字列] デフォルト：'id'	データベーステーブルの主キーになる属性フィールドを定義します
主キー（既存フィールド。上記オプションが空のままの場合に使用） オプション	PRIMARY_KEY	[テーブルのフィールド：任意] デフォルト： なし	データベーステーブルの主キーになるエクスポートレイヤの属性フィールドを定義します
ジオメトリのカラム名 オプション	GEOCOLUMN	[文字列] デフォルト：'geom'	ジオメトリ情報になるデータベースの属性フィールドを定義します

次のページに続く

表 27.271 – 前のページからの続き

ラベル	名前	タイプ	説明
ベクタの次元 オプション	DIM	[列挙型] デフォルト：0 (2D)	インポートされるベクタファイルのデータが 2D か 3D かを定義します。次のいずれかです： • 0 --- 2 • 1 --- 3
簡素化の許容距離 オプション	SIMPLIFY	[文字列] デフォルト： "	インポートされるベクタジオメトリの簡素化のための許容距離を定義します。デフォルトは簡素化しません。
2 地点間の最大距離（高密度化） オプション	SEGMENTIZE	[文字列] デフォルト： "	2 地点間の最大距離。中間点を作成するのに使われます。デフォルトでは高密度化しません。
矩形領域で地物を選択する（入力レイヤの座標系） オプション	SPAT	[範囲] デフォルト： なし	出力テーブルに含まれる地物を範囲指定によって選択します。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算... ：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力
上記の矩形領域で入力レイヤを切り抜く オプション	CLIP	[ブール値] デフォルト： False	先に定義した範囲で入力レイヤを切り抜きます
WHERE 文で地物を選択する (例 column="value") オプション	WHERE	[文字列] デフォルト： "	入力レイヤからどの地物を選択するかを SQL の "WHERE" 文で定義します

次のページに続く

表 27.271 – 前のページからの続き

ラベル	名前	タイプ	説明
トランザクション毎に N 個の地物をグループ化する（デフォルト：2000） オプション	GT	[文字列] デフォルト：''	入力地物を大きさを N で定義したトランザクションにグループ化することができます。デフォルトでは、N はトランザクションの大きさを 20000 地物に制限します。
既存テーブルを上書きする オプション	OVERWRITE	[ブール値] デフォルト：True	データベースに同じ名前のテーブルがあり、このオプションが True に設定されている場合、そのテーブルは上書きされます。
既存テーブルに追加する オプション	APPEND	[ブール値] デフォルト：False	チェック／ True の場合、ベクタデータは既存のテーブルに追加されます。入力レイヤで見つかった新しいフィールドは無視されます。デフォルトでは、新しいテーブルが作成されます。
既存テーブルに新規フィールドを追加する オプション	ADDFIELDS	[ブール値] デフォルト：False	有効にすると、ベクタデータは既存のテーブルに追加され、新しいテーブルは作成されません。入力レイヤで見つかった新しいフィールドがテーブルに追加されます。デフォルトでは、新しいテーブルが作成されます。
カラム／テーブル名を変更しない オプション	LAUNDER	[ブール値] デフォルト：False	このオプションをチェックすると、デフォルトの動作（カラム名を小文字に変換し、空白やその他の無効な文字を削除する）を防ぐことができます。
空間インデックスを作らない オプション	INDEX	[ブール値] デフォルト：False	出力テーブルの空間インデックスが作成されないようにします。デフォルトでは、空間インデックスが追加されます。
失敗した地物はスキップして処理を継続する オプション	SKIPFAILURES	[ブール値] デフォルト：False	
マルチパートに変換する オプション	PROMOTETOMULTI	[ブール値] デフォルト：True	出力テーブルで地物のジオメトリ型をマルチパートにキャストする
入力属性の桁と精度を保持する オプション	PRECISION	[ブール値] デフォルト：True	入力データに従ったカラム属性の変更を行わない
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

このアルゴリズムに出力はありません。

Python コード

Algorithm ID: `gdal:importvectorintopostgisdatabaseavailableconnections`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

PostgreSQL へ出力（新規接続）

PostgreSQL データベース内のベクタレイヤをインポートします。PostGIS データベースへの新しい接続を作成する必要があります。

このアルゴリズムは GDAL ogr2ogr utility から派生したものです。

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	`INPUT`	[ベクタ：任意]	データベースに出力する OGR がサポートしているベクタレイヤ
シェープエンコーディング オプション	`SHAPE_ENCODING`	[文字列] デフォルト：''	データに適用するエンコーディングを設定します

次のページに続く

表 27.272 – 前のページからの続き

ラベル	名前	タイプ	説明
出力のジオメトリ型	GTYPE	[列挙型] デフォルト： 0	出力のジオメトリ型を定義します。次のいずれかです： • 0 --- • 1 --- NONE • 2 --- GEOMETRY • 3 --- POINT • 4 --- LINESTRING • 5 --- POLYGON • 6 --- GEOMETRYCOLLECTION • 7 --- MULTIPOINT • 8 --- MULTIPOLYGON • 9 --- MULTILINESTRING
CRS を出力に割り当てる オプション	A_SRS	[crs] デフォルト： なし	データベーステーブルの出力 CRS を定義します
出力を再投影する CRS オプション	T_SRS	[crs] デフォルト： なし	出力するときにこの CRS に再投影／変換します
CRS を上書きする オプション	S_SRS	[crs] デフォルト： なし	入力レイヤ CRS を上書きします
ホスト オプション	HOST	[文字列] デフォルト：'localhost'	データベースホストの名前
ポート オプション	PORT	[文字列] デフォルト：'5432'	PostgreSQL データベースサーバーがリッスンしているポート番号
ユーザー名 オプション	USER	[文字列] デフォルト： "	データベースへのログインに使うユーザー名
データベース名 オプション	DBNAME	[文字列] デフォルト： "	データベースの名前
パスワード オプション	PASSWORD	[文字列] デフォルト： "	データベースへの接続にユーザー名とともに使うパスワード
スキーマ（スキーマ名） オプション	SCHEMA	[文字列] デフォルト： 'public'	データベーステーブルのスキーマを定義します
テーブル名、空白の場合は入力の名前を使用します オプション	TABLE	[文字列] デフォルト： "	データベースにインポートされるテーブルの名前を定義します。テーブルの名前のデフォルトは、入力ベクタファイルの名前です。
主キー（新規フィールド） オプション	PK	[文字列] デフォルト：'id'	データベーステーブルの主キーになる属性フィールドを定義します

次のページに続く

表 27.272 – 前のページからの続き

ラベル	名前	タイプ	説明
主キー（既存フィールド。上記オプションが空のままの場合に使用） オプション	PRIMARY_KEY	[テーブルのフィールド：任意] デフォルト： なし	データベーステーブルの主キーになるエクスポートレイヤの属性フィールドを定義します
ジオメトリのカラム名 オプション	GEOCOLUMN	[文字列] デフォルト：'geom'	ジオメトリ情報を格納する属性フィールドを定義する
ベクタの次元 オプション	DIM	[列挙型] デフォルト：0 (2D)	インポートされるベクタファイルのデータが 2D か 3D かを定義します。次のいずれかです： • 0 --- 2D • 1 --- 3D
簡素化の許容距離 オプション	SIMPLIFY	[文字列] デフォルト： "	インポートされるベクタジオメトリの簡素化のための許容距離を定義します。デフォルトは簡素化しません。
2 地点間の最大距離（高密度化） オプション	SEGMENTIZE	[文字列] デフォルト： "	2 地点間の最大距離。中間点を作成するのに使われます。デフォルトでは高密度化しません。
矩形領域で地物を選択する（入力レイヤの座標系） オプション	SPAT	[範囲] デフォルト： なし	出力テーブルに含まれる地物を範囲指定によって選択します。 利用できる方法: • レイヤーから計算...: 現在のプロジェクトに読み込まれたレイヤの範囲を使用します。 • レイアウトマップから計算...: アクティブなプロジェクト内の **レイアウト地図アイテム** の範囲を使用します。 • ブックマークから計算...：保存された **ブックマーク** の範囲を使用します。 • 現在のキャンバス領域を使用 • キャンバスに描画: 考慮する領域を区切る矩形をクリック＆ドラッグします。 • `xmin, xmax, ymin, ymax` として座標を入力

次のページに続く

表 27.272 – 前のページからの続き

ラベル	名前	タイプ	説明
上記の矩形領域で入力レイヤを切り抜く オプション	CLIP	[ブール値] デフォルト：False	先に定義した範囲で入力レイヤを切り抜きます
結合するフィールド（すべてのフィールドを結合する場合は空のまま） オプション	FIELDS	[文字列] [リスト] デフォルト：[]	インポートしたベクタファイルから保持するフィールドを定義します。何も選択されていない場合は、すべてのフィールドがインポートされます。
WHERE 文で地物を選択する（例 column="value"） オプション	WHERE	[文字列] デフォルト："	出力テーブルにどの地物を選択するかを SQL の "WHERE" 文で定義します
トランザクション毎に N 個の地物をグループ化する（デフォルト：2000） オプション	GT	[文字列] デフォルト："	入力地物を大きさを N で定義したトランザクションにグループ化することができます。デフォルトでは、N はトランザクションの大きさを 20000 地物に制限します。
既存テーブルを上書きする オプション	OVERWRITE	[ブール値] デフォルト：True	データベースに同じ名前のテーブルがあり、このオプションが True に設定されている場合、そのテーブルは上書きされます。
既存テーブルに追加する オプション	APPEND	[ブール値] デフォルト：False	チェック／True の場合、ベクタデータは既存のテーブルに追加されます。入力レイヤで見つかった新しいフィールドは無視されます。デフォルトでは、新しいテーブルが作成されます。
既存テーブルに新規フィールドを追加する オプション	ADDFIELDS	[ブール値] デフォルト：False	有効にすると、ベクタデータは既存のテーブルに追加され、新しいテーブルは作成されません。入力レイヤで見つかった新しいフィールドがテーブルに追加されます。デフォルトでは、新しいテーブルが作成されます。
カラム／テーブル名を変更しない オプション	LAUNDER	[ブール値] デフォルト：False	このオプションをチェックすると、デフォルトの動作（カラム名を小文字に変換し、空白やその他の無効な文字を削除する）を防ぐことができます。
空間インデックスを作らない オプション	INDEX	[ブール値] デフォルト：False	出力テーブルの空間インデックスが作成されないようにします。デフォルトでは、空間インデックスが追加されます。

次のページに続く

表 27.272 – 前のページからの続き

ラベル	名前	タイプ	説明
失敗した地物はスキップして処理を継続する オプション	SKIPFAILURES	[ブール値] デフォルト：False	
マルチパートに変換する オプション	PROMOTETOMULTI	[ブール値] デフォルト：True	出力テーブルで地物のジオメトリ型をマルチパートにキャストする
入力属性の桁と精度を保持する オプション	PRECISION	[ブール値] デフォルト：True	入力データに従ったカラム属性の変更を行わない
追加の生成オプション オプション	OPTIONS	[文字列] デフォルト：''（追加オプションなし）	追加の GDAL 生成オプション

出力

このアルゴリズムに出力はありません。

Python コード

アルゴリズム ID: `gdal:importvectorintopostgisdatabasenewconnection`

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

ベクタの情報

OGR がサポートするデータ・ソースに関する情報を一覧表示する情報ファイルを作成します。出力は「結果」ウィンドウに表示され、HTML ファイルに書き出すことができます。情報には、ジオメトリ型、地物の数、空間範囲、投影法の情報などが含まれます。

このアルゴリズムは GDAL ogrinfo utility から派生したものです。

パラメータ

ラベル	名前	タイプ	説明
入力レイヤ	INPUT	[ベクタ：任意]	入力ベクタレイヤ
要約の情報のみ オプション	SUMMARY_ONLY	[ブール値] デフォルト： True	
メタデータ情報を出力しない オプション	NO_METADATA	[ブール値] デフォルト： False	
出力ファイル	OUTPUT	[html] デフォルト：[一時ファイルに保存]	ファイル情報を含む出力 HTML ファイルを指定します。次のいずれかです： • 一時ファイルに保存 • ファイルに保存... HTML ファイルが指定されないときは、一時ファイルに出力が書き込まれます

出力

ラベル	名前	タイプ	説明
出力ファイル	OUTPUT	[html]	ファイル情報を含む出力 HTML ファイル

Python コード

アルゴリズム ID: gdal:ogrinfo

```
import processing
processing.run("algorithm_id", {parameter_dictionary})
```

algorithm id は、プロセシングツールボックス内でアルゴリズムにマウスカーソルを乗せた際に表示されます。 *parameter dictionary* は、パラメータの「名前」とその値を与えます。Python コンソールからプロセシングアルゴリズムを実行する方法の詳細については、 **プロセシングアルゴリズムをコンソールから使う** を参照してください。

27.3 OTB アプリケーションプロバイダ

OTB (Orfeo ToolBox) は、リモートセンシングデータ用の画像処理ライブラリです。また、画像処理機能を提供するアプリケーションも提供されています。アプリケーションの一覧とそのドキュメントは OTB CookBook に掲載されています

第28章　プラグイン

28.1 QGIS プラグイン

QGIS は、プラグインアーキテクチャで設計されています。これは、多くの新機能や機能を簡単にアプリケーションに追加することを可能にします。実際に QGIS の機能のいくつかはプラグインとして実装されています。

28.1.1 コアプラグインと外部プラグイン

QGIS プラグインは **コア・プラグイン** または **外部プラグイン** として実装されます。

コア・プラグイン は QGIS 開発チームによって維持管理され、自動的にすべての QGIS ディストリビューションの一部になっています。それらは **C++** と **Python** という 2 つの言語の、いずれかで書かれています。

外部プラグインのほとんどは、現在 Python で書かれています。これらは、https://plugins.qgis.org/plugins/ の「公式」QGIS リポジトリ、または外部リポジトリに保存され、個々の作者によって管理されています。公式リポジトリにあるプラグインについては、使用方法、QGIS の最小バージョン、ホームページ、作者、その他の重要な情報についての詳細な文書が提供されています。その他の外部リポジトリについては、外部プラグイン自体にドキュメントが用意されている場合があります。外部プラグインのドキュメントは、本マニュアルには含まれていません。

プラグインをインストールまたは有効化するには、 **プラグイン** メニューから **プラグインの管理とインストール**... を選択してください。インストールされた外部 Python プラグインは、アクティブな *user profile* パスの `python/plugins` フォルダの下に配置されます。

カスタム C ++のプラグインライブラリへのパスも **設定**　**オプション**　**システム** 下に追加できます。

28.1.2 プラグインダイアログ

[設定] タブ

左パネルの下部にある **設定** タブは、アプリケーションで表示できるプラグインを設定する主な場所です。以下のオプションを使用することができます:

- **起動時に更新を確認する**。インストールされているプラグインにアップデートがある場合、QGIS は *QGIS* **起動時**、*1* **日** *1* **回**, *3* **日に** *1* **回**, **毎週**、 *2* **週間毎** または **毎月** に通知します。

- **実験的プラグインも表示**。QGIS では、一般的に実運用には適さない開発初期段階のプラグインを表示します。これらのプラグインについては、安定版と実験版のどちらかをインストールし、いつでも切り替えることができます。
- **非推奨プラグインも表示**。これらのプラグインは、QGIS の機能を代替している、メンテナが不足している、QGIS で利用できなくなった機能に依存している... などの理由から、通常メンテナンスされていません。これらのプラグインは一般的に実運用には適さず、プラグインリストでは灰色で表示されます。

デフォルトでは、**プラグインリポジトリ** セクションで、QGIS は URL `https://plugins.qgis.org/plugins/plugins.xml?qgis=version` （ここで `<version>` はあなたが実行している正確な QGIS バージョンを示します）を持つ公式プラグインリポジトリを提供します。外部の作者のリポジトリを追加するには、 **追加**... をクリックし、 **リポジトリの詳細** フォームに名前と URL を入力します。URL は `http://` または `file://` プロトコルタイプになります。

デフォルトの QGIS リポジトリはオープンなリポジトリで、アクセスするための認証は必要ありません。ただし、独自のプラグインリポジトリを展開して認証（基本認証、PKI）を必要にすることができます。 QGIS 認証のサポートに関する詳細は 認証 の章を参照してください。

追加されたリポジトリのうち 1 つ以上が不要な場合は、設定タブから **編集**... ボタンを使って無効にしたり、 **削除** ボタンを使って完全に削除することができます。

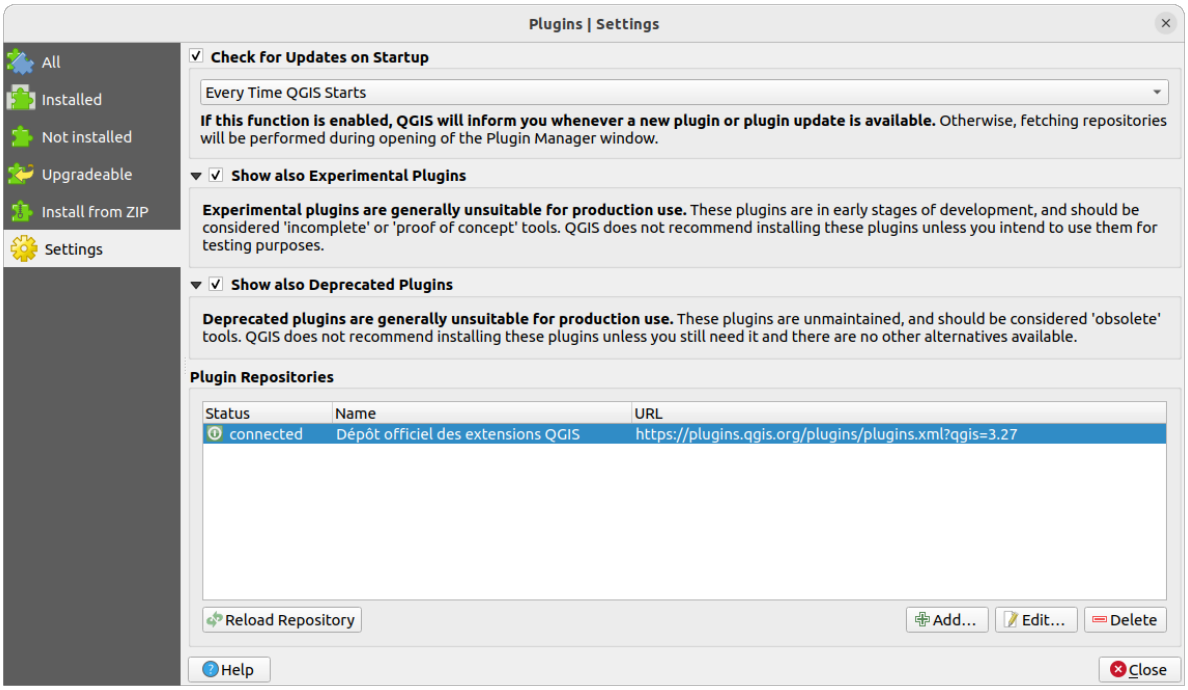


図 28.1: **設定** タブ

プラグインを眺める

タブ

プラグイン ダイアログの上部のタブは、インストール、作成、更新のステータスに基づいたプラグインのリストを提供します。プラグインの設定により、利用可能なタブは以下の通りです:

- **すべて**: 有効なリポジトリにある利用可能なすべてのプラグインを表示します
- **インストール済**: インストールしたプラグインと、デフォルトでインストールされアンインストールできないコアプラグインの両方を表示します
- **未インストール**: 有効なリポジトリにある、アンインストールまたは未だインストールされていないプラグインを表示します
- **新規**: 前回の **起動時に更新を確認する** からリリースされたプラグインを表示します
- *Upgradeable*: インストールされているプラグインのうち、リポジトリでより新しいバージョンを公開しているものを表示します
- *Invalid*: インストールされているプラグインのうち、何らかの理由で現在壊れているもの（依存関係がない、読み込み中にエラーが出る、QGIS バージョンと互換性のない機能...）をすべて表示します

タブの上部には **検索** 機能があり、メタデータ情報（作者、名前、説明、タグ、...）から任意のプラグインを探すことができます。

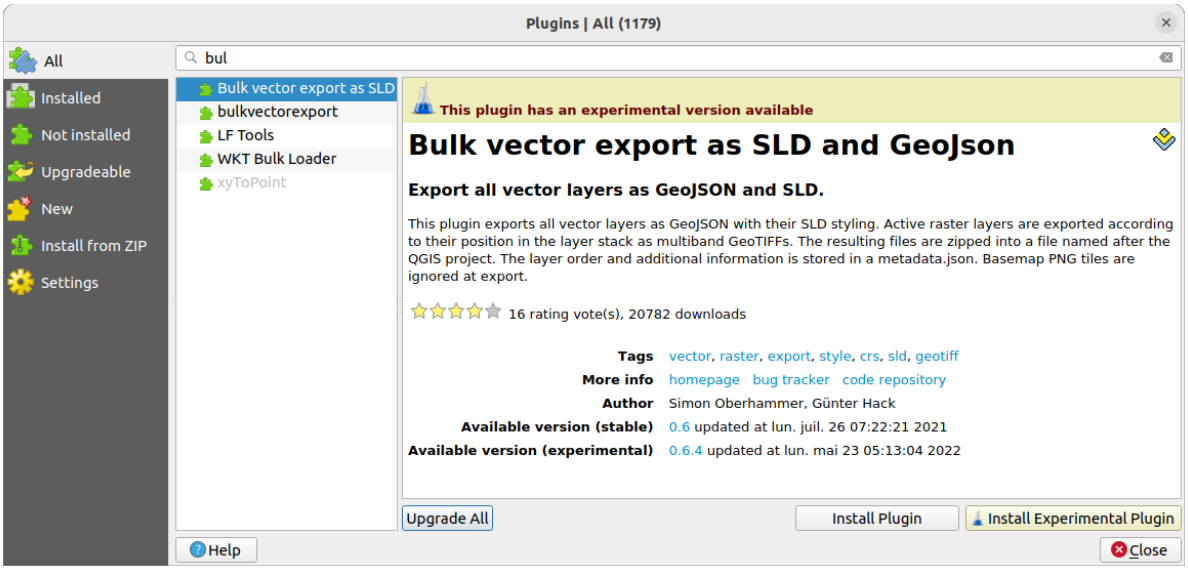


図 28.2: **すべて** タブからプラグインを検索する

プラグイン

プラグインを選択すると、右のパネルにいくつかのメタデータが表示されます:

- プラグインが実験的であるか、実験版が利用可能であるかについての情報（ **実験的プラグインも表示** がチェックされている場合）
- サマリと説明
- 評価投票（お好みのプラグインに投票できます！）
- タグ
- ホームページ、トラッカーやコードリポジトリにはいくつかの便利なリンク
- 作者
- 利用可能なバージョンと、リポジトリ内のダウンロードページへのリンク、またはインストールされたプラグインのローカルフォルダへのパス

プラグインマネージャ ダイアログでは、最新バージョンのプラグインを操作することができます。有効にすると、実験版は、最新の安定版よりも新しい場合にのみ表示されます。アクティブなタブに応じて、選択したプラグインがインストールされているかどうか、以下のオプションのいくつかが表示されます:

- **インストール**: 選択したプラグインの最新の安定版をインストールします
- **実験的なプラグインをインストール**: 選択したプラグインの実験版をインストールします
- **再インストール**: 同じ安定したバージョンのプラグインをインストールします 例: ロードに失敗した後
- **実験的なプラグインを再インストール**: 同じ安定版のプラグインをインストールします 例: ロードに失敗した後
- **プラグインをアップグレード**: 選択したプラグインを最新の安定版にアップグレードします
- **実験的なプラグインをアップグレード**: 選択したプラグインを実験版にアップグレードします
- **すべてアップグレード**: インストールされているすべてのプラグインを、より新しい安定版または実験版にアップグレードします（以前にインストールされていたバージョンが安定版か実験版かによって異なります）。
- *Downgrade Plugin*: プラグインの実験版から、以前の安定版に移行します
- *Downgrade Experimental Plugin*: プラグインの実験版から、その最新の公開済みの実験版に移動します。これは、まだ公開されていないバージョンで遊んでいるときに発生する可能性があります。
- **アンインストール**: インストールされたプラグインをユーザープロファイルから削除します

インストールされたプラグインは、その左側に チェックボックスが表示されます。チェックを外すと、一時的にプラグインを停止することができます。

リスト内のプラグインを右クリックすると、プラグインリストを様々なメタデータでソートできるようになります。新しい順序は、すべてのタブに適用されます。ソートオプションは以下の通りです:

- **名前で並び替え**

- **ダウンロード数で並べ替え**
- **投票数で並べ替え**
- **状態で並べ替え**
- **作成日時で並べ替え**
- **更新日時で並べ替え**

ZIP からインストールタブ

ZIP **からインストール** タブは、リポジトリから直接ダウンロードしたプラグインなど、zip 形式のプラグインをインポートするためのファイルセレクタウィジェットを提供します。暗号化されたファイルもサポートされています。

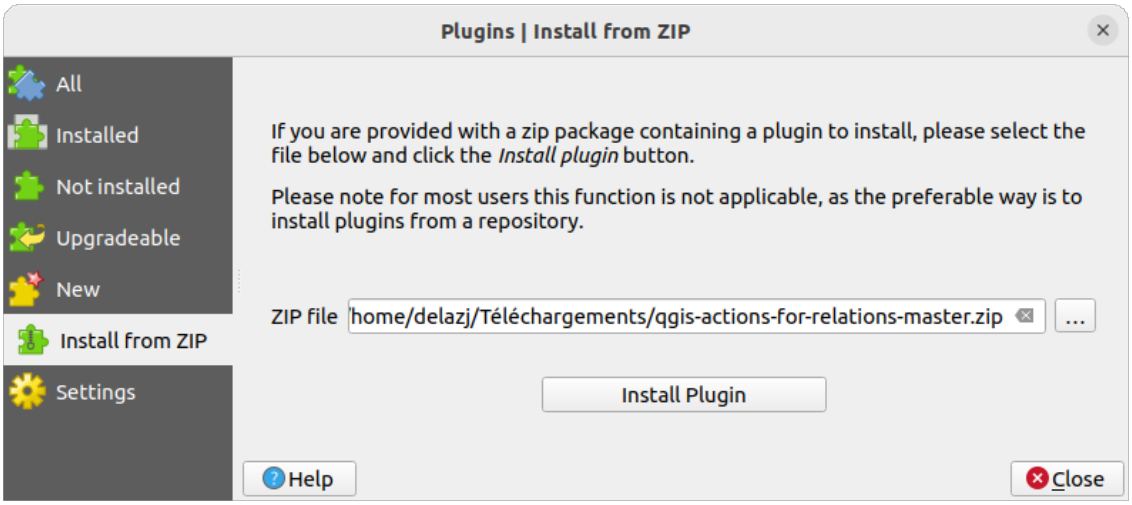


図 28.3: installPluginFromZip|:guilabel:*zip* **からインストール** タブ

28.2 QGIS コア・プラグインを使用する

28.2.1 DB マネージャプラグイン

DB マネージャプラグインは、QGIS でサポートされている空間データベース形式（PostGIS、SpatiaLite、GeoPackage、Oracle Spatial、Virtual layers）を一つのユーザーインターフェースで統合し管理するメインツールとなることを意図しています。DB マネージャプラグイン はいくつかの機能を提供します。QGIS ブラウザから DB マネージャにレイヤをドラッグすると、レイヤを空間データベースにインポートすることができます。空間データベース間でテーブルをドラッグ＆ドロップすることができ、それらはインポートされます。

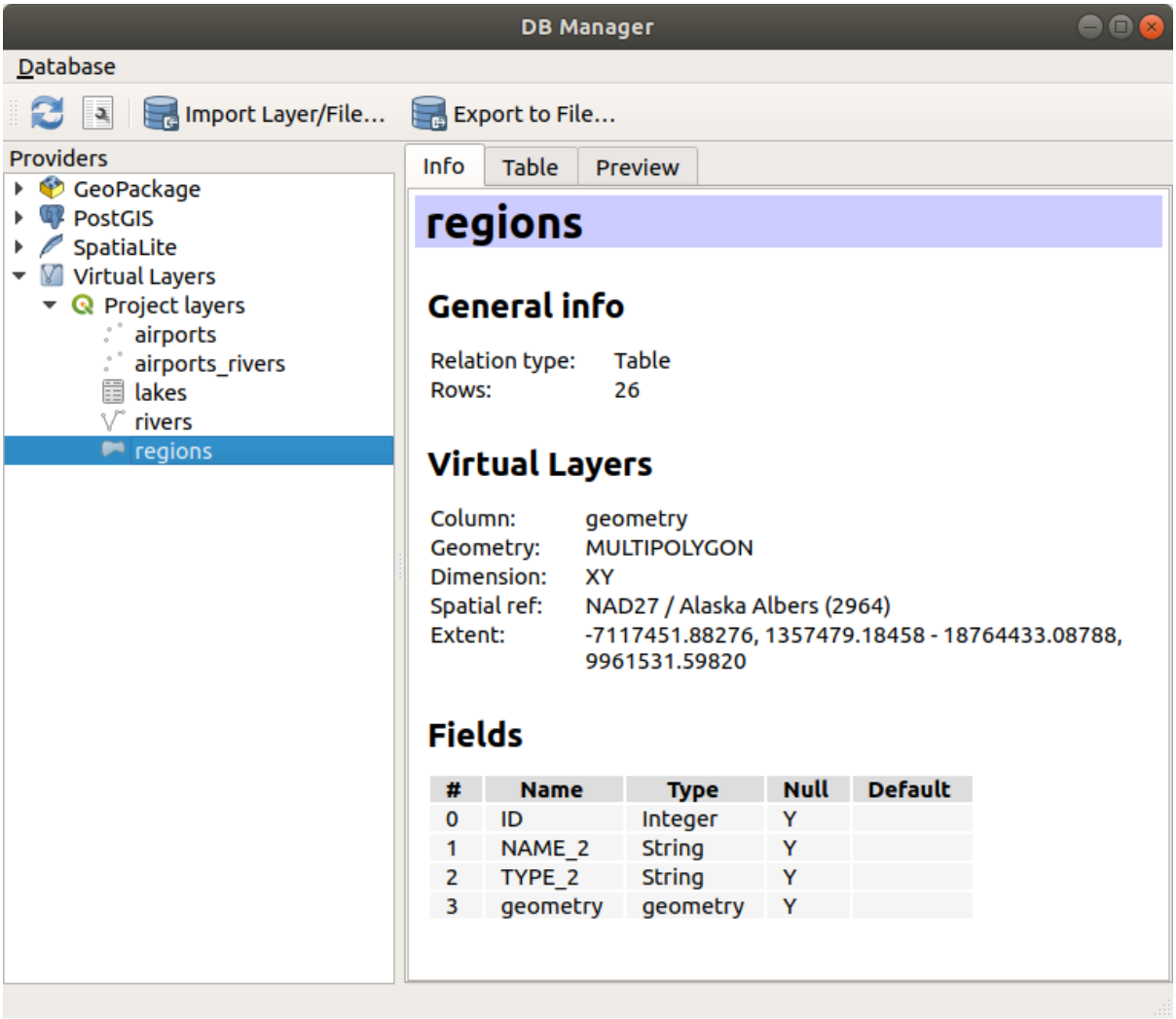


図 28.4: [DB マネージャ] ダイアログ

データベース メニューでは、既存のデータベースへの接続、SQL ウィンドウの起動、DB マネージャプラグインの終了を行うことができます。既存のデータベースに接続すると、 **スキーマ** (PostGIS / PostgreSQL などの DBMS に関連) と **テーブル** というメニューが表示されます。

スキーマ メニューにはスキーマの作成と削除（空の場合のみ)、そしてトポロジが利用可能な場合（PostGIS トポロジなど）は、 *TopoViewer* を起動するツールがあります。

テーブル メニューではテーブルの作成と編集、テーブルとビューの削除ができます。また、テーブルを空にしたり、スキーマ間でテーブルを移動することもできます。選択したテーブルに対して **バキューム解析の実行** をすることができます。バキュームはスペースを回収して最良できるようにし、**解析** はクエリの最も効率的な実行方法を決めるために使われる統計情報を更新します。*guilabel:**変更ログ**... では、テーブルに変更ログのサポートを追加することができます。最後に、 **レイヤ/ファイルのインポート**... と **ファイルにエクスポート**... ができます。

注釈: DB Manager を使うと、PostgreSQL データベースのテーブルやカラムにコメントを追加することができます。

プロバイダ ウィンドウには、QGIS でサポートされている既存のデータベースが一覧表示されます。ダブルクリックするとそのデータベースに接続できます。マウスの右ボタンで、既存のスキーマやテーブルの名前の変更と削除ができます。テーブルはコンテキストメニューで QGIS キャンバスに追加することもできます。

データベースに接続されている場合、DB マネージャの **メイン** ウィンドウには 4 つのタブが表示されます。**情報** タブはテーブルとそのジオメトリ、既存のフィールド、制約、インデックスに関する情報を提供します。選択したテーブルに空間インデックスを作成することができます。**テーブル** タブはテーブルを表示し、**プレビュー** タブはジオメトリをプレビューとしてレンダリングします。*SQL* **ウィンドウ** を開くと、新しいタブが表示されます。

SQL ウィンドウの操作

DB マネージャを使って空間データベースに対する SQL クエリが実行できます。クエリは保存と読み込みが可能で、*SQL* **クエリビルダ** がクエリの作成を手助けしてくれます。**新規レイヤとして読み込む** にチェックを入れ、 **一意な値を持つカラム** (ID)、**ジオメトリカラム**、 **レイヤ名** *(***接頭辞***)* を指定することで、空間的な出力を表示することもできます。Ctrl+R を押すか、**実行** ボタンをクリックすると、SQL の一部をハイライトしてその部分のみを実行することができます。

クエリ履歴 ボタンは、各データベースとプロバイダの直近 20 回のクエリを保存しています。

エントリをダブルクリックすると、その文字列が SQL ウィンドウに追加されます。

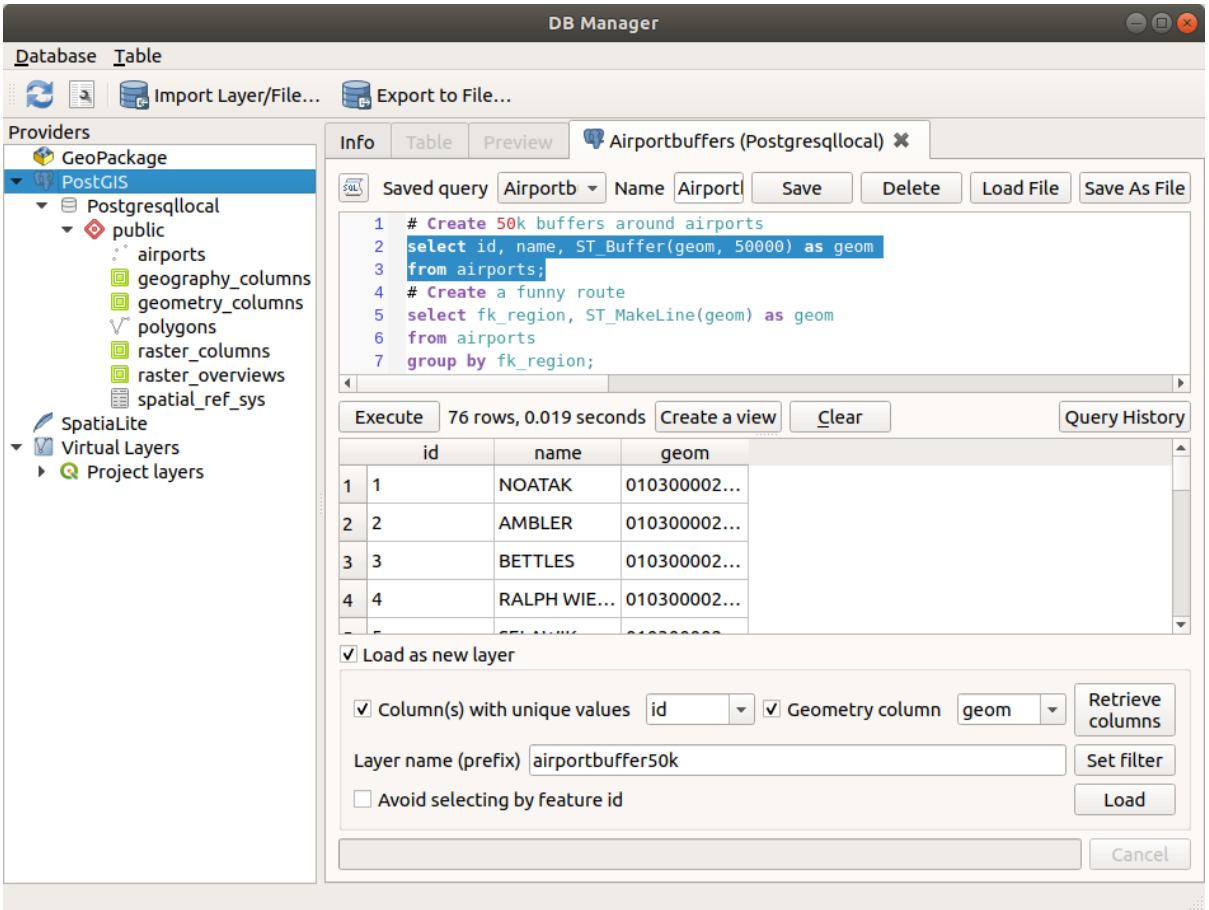


図 28.5: DB マネージャの SQL ウィンドウで SQL クエリを実行する

注釈: SQL ウィンドウから仮想レイヤを作ることもできます。その場合はデータベースを選択する代わりに、SQL ウィンドウを開く前に **仮想レイヤ** の下にある **QGIS レイヤ** を選択します。使用する SQL 構文については **仮想レイヤを作成する** を参照してください。

28.2.2 ジオメトリチェッカープラグイン

ジオメトリチェッカーは、レイヤのジオメトリの妥当性をチェックして修正するための強力なコアプラグインです。これは **ベクタ** メニューから利用できます（ **ジオメトリをチェック**...）。

チェックを設定する

ジオメトリをチェック ダイアログでは、最初のタブ（**セットアップ**）にさまざまな設定がグループで表示されます:

- **入力ベクタレイヤ**: チェックするレイヤを選択します。 **選択した地物のみ** チェックボックスを使うと、選択された地物のジオメトリにチェックを制限することができます。
- **許容されるジオメトリ型** は入力レイヤのジオメトリ型を制限することができます：
  - 点
  - マルチポイント
  - ライン
  - マルチライン
  - ポリゴン
  - マルチポリゴン
- **ジオメトリの有効性**。ジオメトリ型によって次が選べます:
  - **自己交差**
  - **重複頂点**
  - **自己接触**
  - **頂点が *3* つより少ないポリゴン**。
- **ジオメトリプロパティ**。ジオメトリ型によって、利用できるオプションが異なります：
  - **ポリゴンに穴がない**
  - **マルチパートの地物は必ず複数のジオメトリ**
  - **線にダングル（懸垂線）があってはならない**
- **ジオメトリの条件**。ジオメトリを検証するための条件を追加できます：
  - **最短セグメント長（地図単位）** 1,00

  - **セグメント間の最小角度（度）** 1,00
  - **最小ポリゴン領域（地図単位）** 1,00
  - **最大の細さ** 1,00 と **最大面積（地図単位）** 1,00 を持った **極細ポリゴン** *(sliver)* **がない**
- **トポロジチェック**。ジオメトリ型によって異なる、たくさんのオプションが利用できます:
  - **重複ジオメトリ**
  - **地物の内部に他の地物がない**
  - **重なりがこの値より小さい（地図単位）** 1,00
  - **隙間がこの値より小さい（地図単位）** 1,00
  - **点は線のノードに重ならなければならない**
  - **点はポリゴンの内側になければならない**
  - **線は他の線と交差してはならない**
  - **線は右のレイヤの地物と交差してはならない**
  - **ポリゴンは右のレイヤの範囲内になければならない**
- **許容範囲**。マップレイヤの単位でチェックの許容範囲を定義できます。
- **出力ベクタレイヤ** には次の選択肢があります:
  - **入力レイヤを変更**
  - **新規レイヤを作成**

設定に満足したら **実行** ボタンをクリックします。

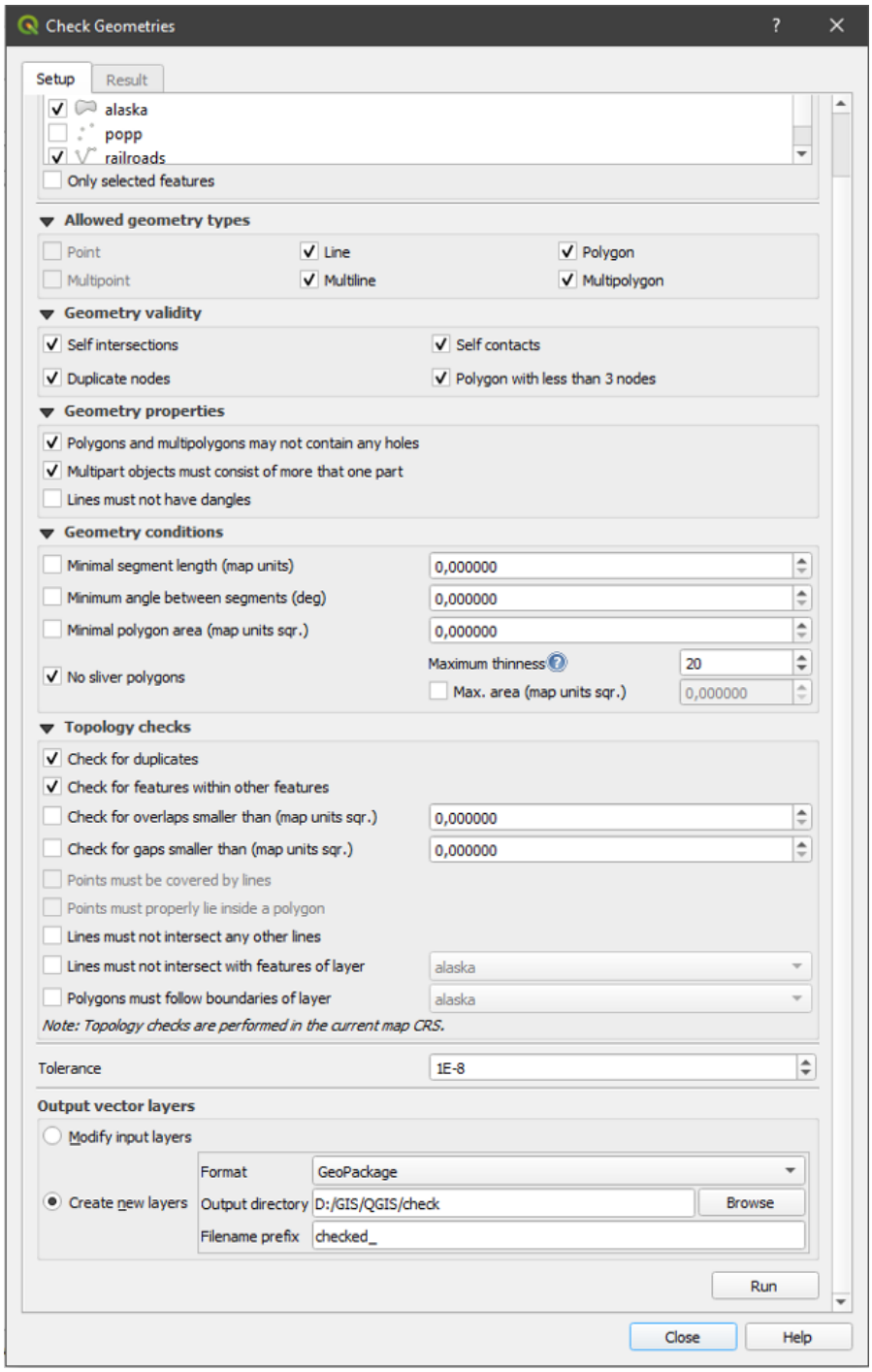


図 28.6: ジオメトリチェッカープラグイン

ジオメトリチェッカープラグイン は次のエラーを見つけることができます:

- 自己交差: 自己交差しているポリゴン
- 重複頂点: ひとつのセグメントに二つの重複した頂点
- 穴: ポリゴンの穴
- セグメント長: しきい値よりも短いセグメント長
- 最小角度: しきい値よりも浅い角度を持った二つのセグメント

- 最小面積: しきい値よりも狭いポリゴンの面積
- スライバポリゴン: このエラーは、長い周長を持った非常に小さな（面積が小さな）ポリゴンの場合に発生します
- 重複した地物
- 地物の内部にある地物
- 重なり: ポリゴンの重なり合い
- 隙間: ポリゴンの間の隙間

次の図は、プラグインが行うさまざまなチェックを示しています。

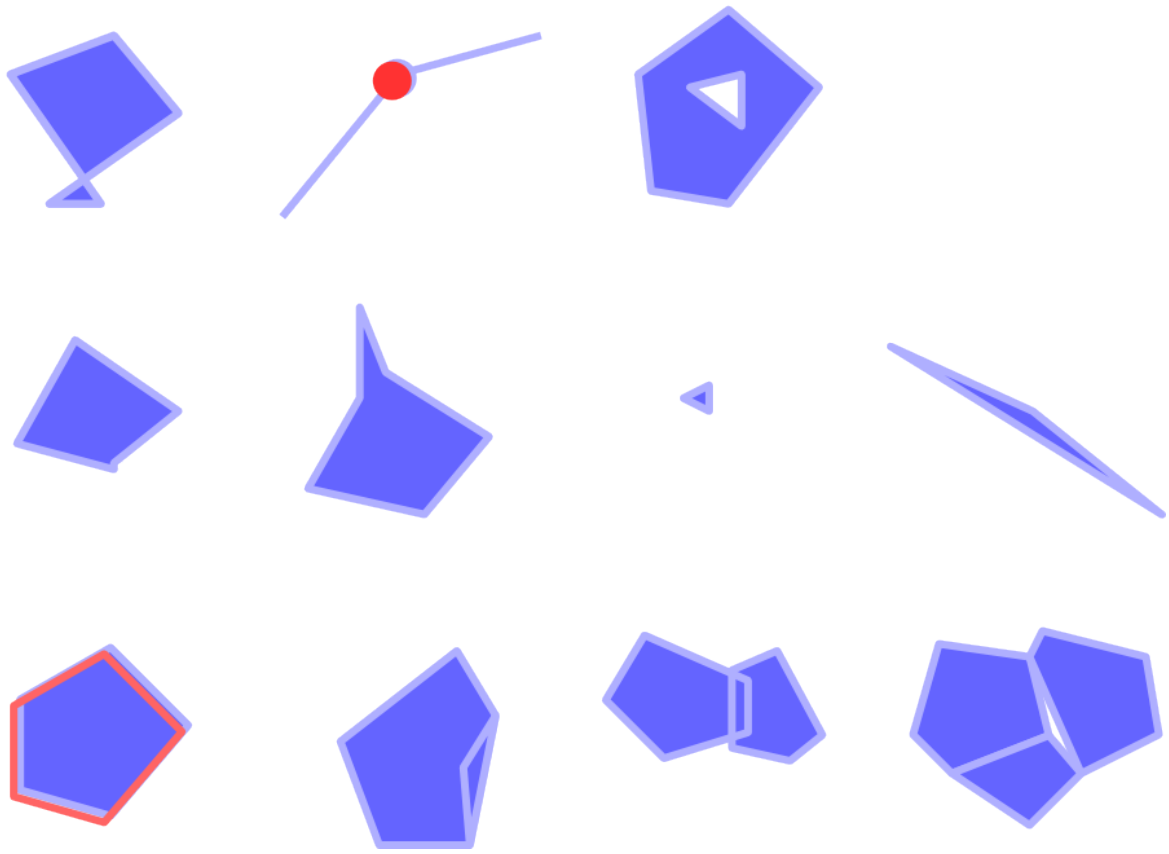


図 28.7: プラグインがサポートするチェックの一部

結果を分析する

結果は 2 番目のタブ (**出力**) に表示され、キャンバスにエラーの概要レイヤとして表示されます (その名前はデフォルトの接頭辞 `checked_` を持ちます)。テーブルには **ジオメトリチェックの結果** が一覧表示され、1 行に 1 つのエラー、列にはレイヤ名、ID、エラータイプ、エラーの座標、値（エラーのタイプによる）、そして最後にエラーの解決方法を示す解決列が表示されます。このテーブルの下部には、エラーをさまざまなファイル形式に **エクスポート** することができます。また、総エラー数と修正済みエラー数のカウンタもあります。

行を選択してエラーの場所を見ることができます。この動作は **エラー**（デフォルト）、**地物**、**移動しない**、**選択地物を強調** の中から別の動作を選択することで変更できます。

テーブル行をクリックしたときのズーム操作の下では、次をすることができます：

- **選択した地物を属性テーブルで表示**
- **選択したエラーをデフォルトの解決方法で修正する**
- **選択したエラーを修正し、解決方法を尋ねる** 解決の方法を選択するウィンドウが表示されます:
  - 最も長い共有エッジの隣接ポリゴンをマージする
  - 最大面積の隣接ポリゴンをマージする
  - 同一の属性値を持つ隣接ポリゴンをマージする。もしなければそのままにする
  - 地物を削除
  - 何もしない
- **エラーの解決方法の設定** では、エラーの種類に応じてデフォルトの解決方法を変更することができます

Tip: 複数のエラーの修正

CTRL+**クリック** アクションでテーブル内の複数の行を選択し、複数のエラーを修正することができます。

最後に、**属性値で地物をマージする際に使用する属性** を選択することができます。

28.2.3 MetaSearch Catalog Client

はじめに

MetaSearch は、メタデータカタログサービスと対話するための QGIS プラグインで、OGC API - Records と OGC Catalog Service for the Web (CSW) 標準の両方をサポートしています。

MetaSearch は、QGIS 内のメタデータカタログを検索するための簡単で直感的なアプローチとユーザーフレンドリーなインターフェースを提供します。

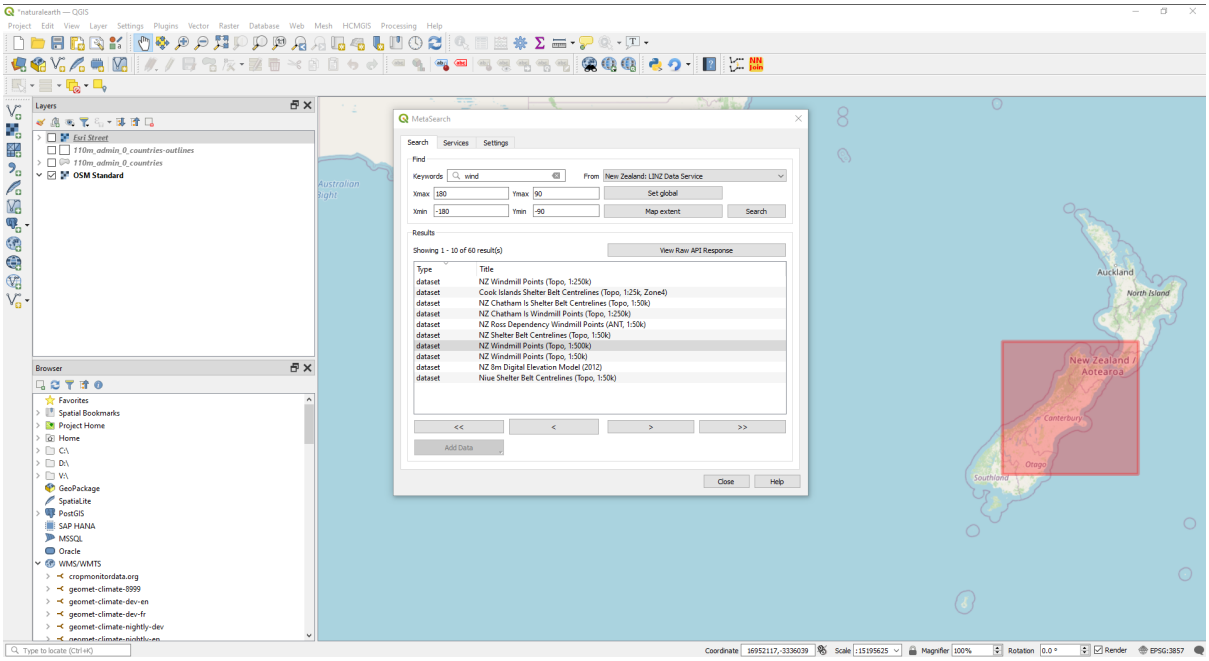


図 28.8: MetaSearch におけるサービスの検索と結果

QGIS でメタデータカタログを操作する

MetaSearch とそれが必要とするものは QGIS にデフォルトで含まれており、QGIS プラグインマネージャから有効にすることができます。

OGC API - Records

OGC API - Records`_ は、OGC (Open Geospatial Consortium)`_ の標準規格で、ウェブ上の地理空間リソースを発見するためのものです。

CSW (Catalog Service for the Web)

CSW（Catalog Service for the Web）`_は OGC（Open Geospatial Consortium）`_ の仕様で、データ、サービス、その他の潜在的なリソースに関するメタデータを発見、閲覧、照会するための共通インターフェースを定義しています。

始める

MetaSearch を始めるには、  アイコンをクリックするか、QGIS のメインメニューから *Web* *MetaSearch* *MetaSearch* を選択してください。メタサーチダイアログが表示されます。メイン GUI は 3 つのタブで構成されています: **サービス**、 **検索**、 **設定**。

カタログサービスを管理する

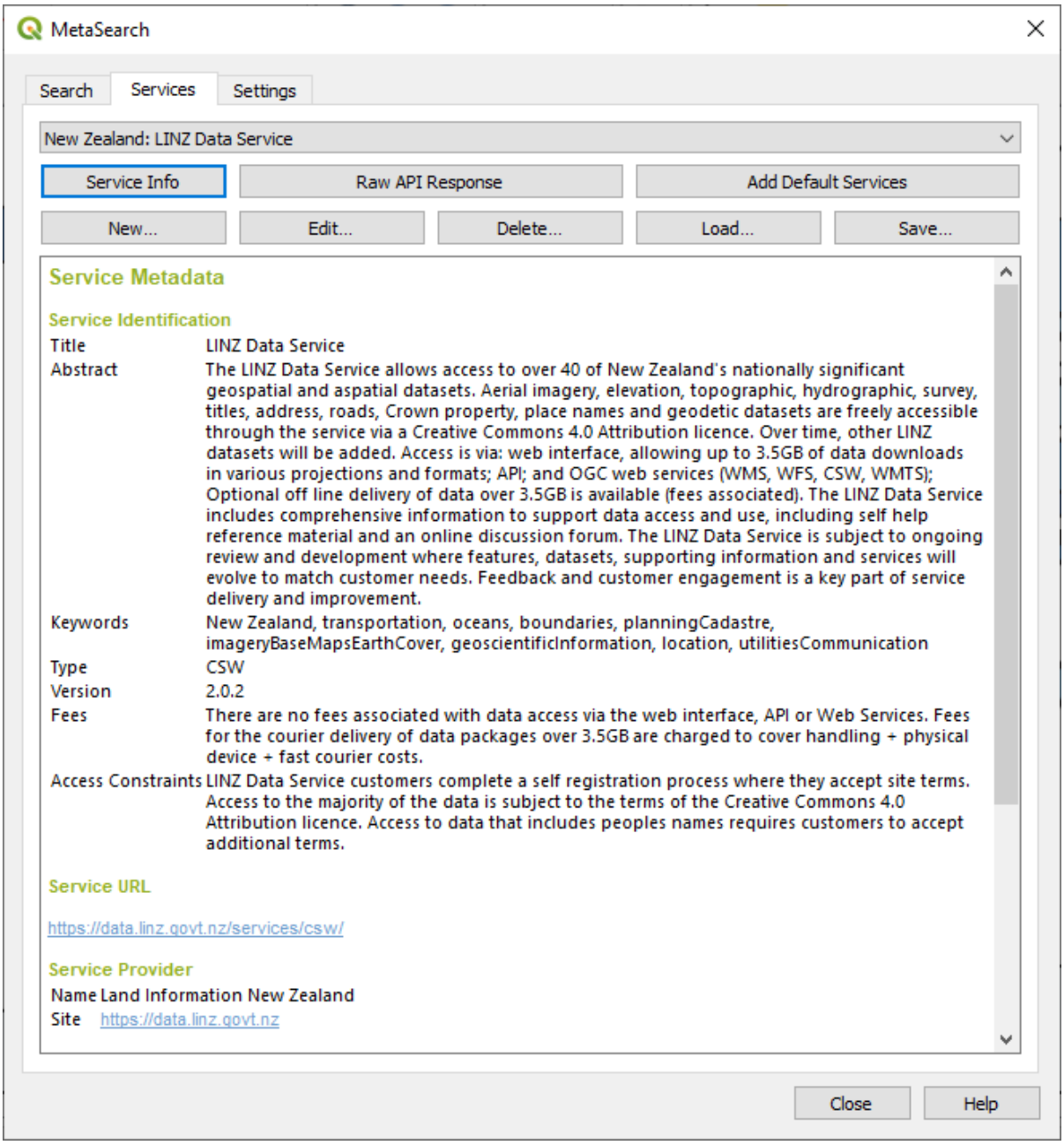


図 28.9: カタログサービスを管理する

サービス タブでは、利用可能なすべてのカタログサービスを管理することができます。MetaSearch にはカタログサービスのデフォルトリストが用意されており、**デフォルトサービスを追加** ボタンを押すことで追加することができます。

リストされたすべてのカタログサービスのエントリを検索するには、ドロップダウンセレクトボックスをクリックします。

カタログサービスのエントリを追加する：

1. **新規** ボタンをクリックする

2. サービスの **名前** と :guilabel:`URL`（エンドポイント）を入力します。OGC CSW 2.0.2 カタログの場合、ベース URL のみが必要であることに注意すること（完全な GetCapabilities URL ではない）。OGC API - Records カタログの場合、URL はコレクション・エンドポイントへのパスでなければならない。
3. カタログに認証が必要な場合は、適切な **ユーザー名** と **パスワード** を入力してください。
4. guilabel:*OK* をクリックして、サービスをエントリのリストに追加します。

既存のカタログサービスのエントリを編集する：

1. 編集したい園地路を選択する
2. **編集** ボタンをクリックする
3. **名前** または *URL* の値を変更する
4. *OK* をクリックします。

カタログサービスのエントリを削除するには、削除したいエントリを選択して **削除** ボタンをクリックします。エントリの削除を確認するメッセージが表示されます。

MetaSearch では、XML ファイルを使った接続の読み込みと保存が可能です。これは、アプリケーション間で設定を共有する必要がある場合に便利です。以下は XML ファイル形式の例です。

```
<?xml version="1.0" encoding="UTF-8"?>
<qgsCSWConnections version="1.0">
    <csw type="OGC CSW 2.0.2" name="Data.gov CSW" url="https://catalog.data.gov/csw-
↪all"/>
    <csw type="OGC CSW 2.0.2" name="Geonorge - National CSW service for Norway" url=
↪"https://www.geonorge.no/geonetwork/srv/eng/csw"/>
    <csw type="OGC CSW 2.0.2" name="Geoportale Nazionale - Servizio di ricerca␣
↪Italiano" url="http://www.pcn.minambiente.it/geoportal/csw"/>
    <csw type="OGC CSW 2.0.2" name="LINZ Data Service" url="http://data.linz.govt.nz/
↪feeds/csw"/>
    <csw type="OGC CSW 2.0.2" name="Nationaal Georegister (Nederland)" url="http://
↪www.nationaalgeoregister.nl/geonetwork/srv/eng/csw"/>
    <csw type="OGC CSW 2.0.2" name="RNDT - Repertorio Nazionale dei Dati Territoriali␣
↪- Servizio di ricerca" url="http://www.rndt.gov.it/RNDT/CSW"/>
    <csw type="OGC CSW 2.0.2" name="UK Location Catalogue Publishing Service" url=
↪"http://csw.data.gov.uk/geonetwork/srv/en/csw"/>
    <csw type="OGC CSW 2.0.2" name="UNEP/GRID-Geneva Metadata Catalog" url="http://
↪metadata.grid.unep.ch:8080/geonetwork/srv/eng/csw"/>
</qgsCSWConnections>
```

エントリのリストを読み込む:

1. **読み込み** ボタンをクリックします。新しいウィンドウが表示されます。
2. **ブラウズ** ボタンをクリックし、読み込みたいエントリの XML ファイルに移動します。
3. **開く** をクリックします。エントリのリストが表示されます。

4. リストから追加したいエントリを選択し、 **読み込み** をクリックします。

サービス情報 ボタンをクリックすると、サービスの識別情報、サービスプロバイダ、連絡先情報など、選択したカタログサービスに関する情報が表示されます。生の API レスポンスを表示したい場合は、*Raw API* **レスポンス** ボタンをクリックします。別ウィンドウが開き、生の JSON または XML 形式でサーバー情報が表示されます。

カタログサービスを検索する

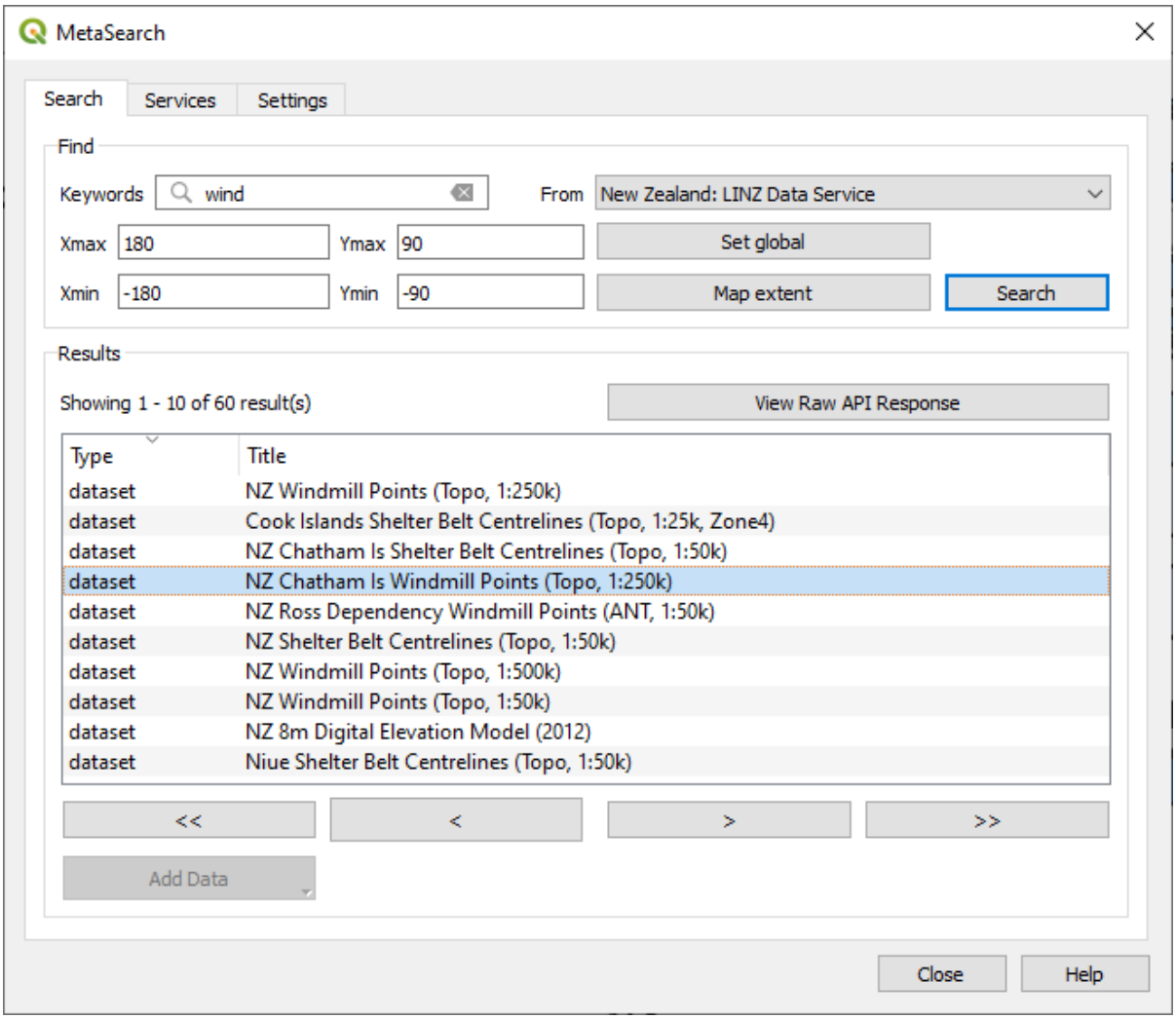


図 28.10: カタログサービスを検索する

検索 タブでは、カタログサービスにデータやサービスを照会したり、さまざまな検索パラメーターを設定したり、結果を表示したりすることができます。

次の検索パラメータが利用可能です:

- **キーワード**: 自由なテキスト検索キーワード;
- **サービス**: クエリを実行するカタログサービス;

- **バウンディングボックス**: フィルタリングする空間領域であり、*X* **最大値**、*X* **最小値**、 *Y* **最大値**、 *Y* **最小値** で定義されます。**グローバルに設定** をクリックしてグローバル検索を行う、**地図の領域** をクリックして可視領域を検索する、または手動で値を入力します。

guilabel:**検索** ボタンをクリックすると、選択したメタデータカタログを検索します。検索結果はリストで表示され、列のヘッダをクリックすることでソートすることができます。検索結果の下にある方向ボタンで検索結果を行き来できます。

結果を選択します:

- *RAW API* **レスポンスの表示** ボタンをクリックすると、生の JSON または XML 形式のサービス応答を表示するウィンドウが開きます。
- メタデータレコードに関連するバウンディングボックスがある場合は、バウンディングボックスのフットプリントが地図上に表示されます。
- レコードをダブルクリックすると、関連するアクセス リンクを含むレコード メタデータが表示されます。リンクをクリックすると、ユーザーのウェブブラウザでリンクが開きます。
- レコードがサポートされているウェブサービス（WMS/WMTS、WFS、WCS、ArcGIS REST Service など）である場合、**データを追加** ボタンが有効になります。このボタンをクリックすると、MetaSearch はこれが有効な OWS かどうかを確認します。次にサービスが適切な QGIS 接続リストに追加され、適切な接続ダイアログが表示されます。

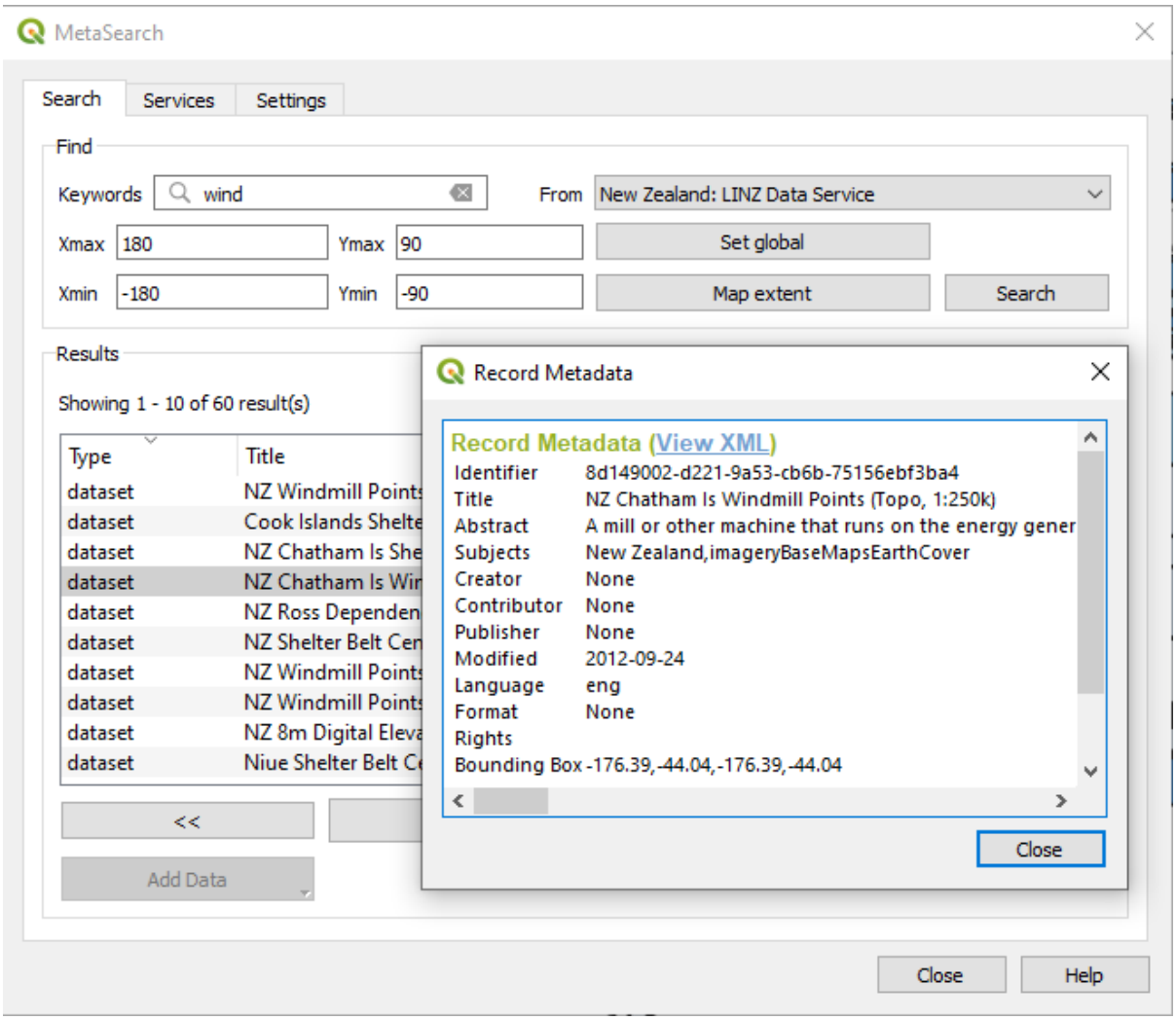


図 28.11: メタデータレコードの表示

設定

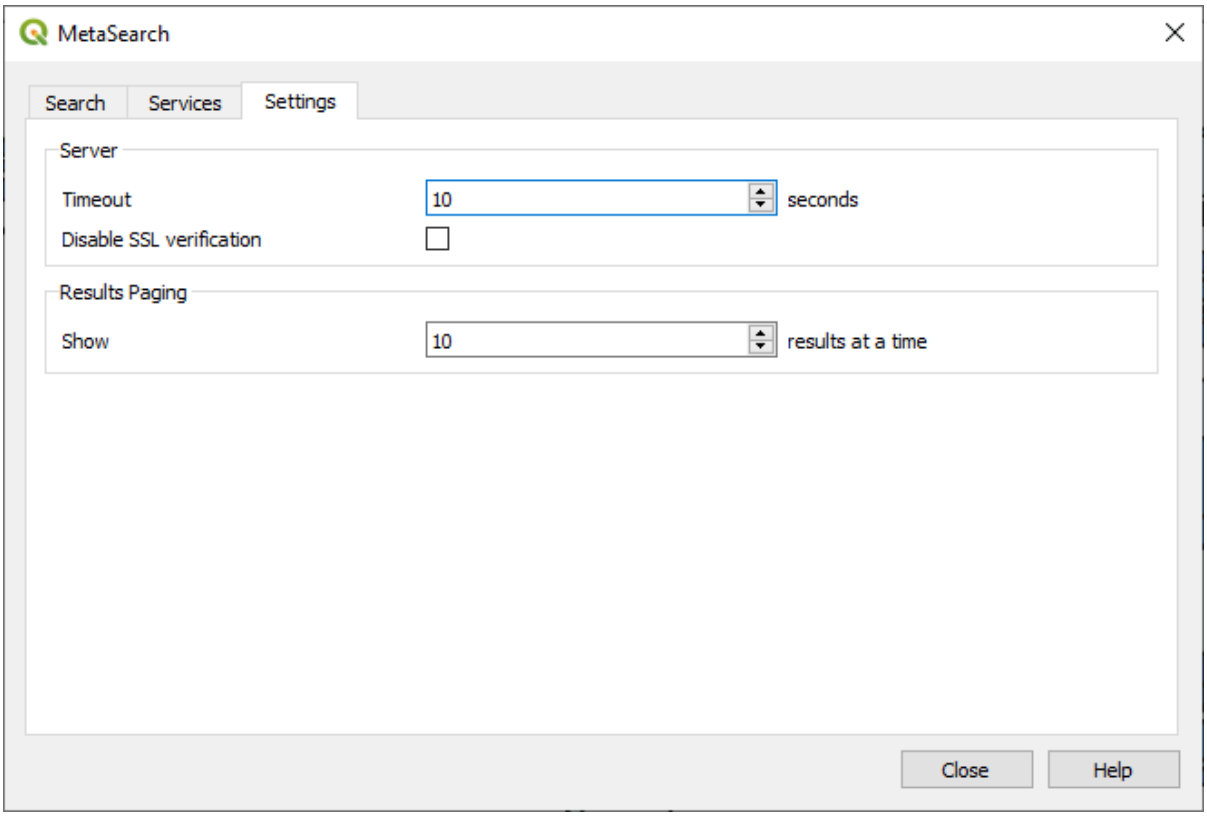


図 28.12: メタサーチの設定

MetaSearch は以下の:guilabel:**設定** で微調整できます：

- **タイムアウト**: メタデータカタログを検索する際に、接続試行をブロックする秒数を指定します。デフォルト値は 10。
- *SSL* **認証を無効化**: SSL 検証をオフにするオプション。
- **結果のページング**: メタデータカタログを検索する際に、1 ページに表示する結果の数を指定します。デフォルト値は 10。

カタログサーバーのエラー

カタログがウェブブラウザでは動作するが、MetaSearch では動作しない場合があります。これは、カタログサーバーのコンフィギュレーション/セットアップが原因である可能性があります。カタログサーバープロバイダは、URL の一貫性と最新の設定を確認する必要があります（これは HTTP -> HTTPS のリダイレクトシナリオで一般的です）。この問題と修正についての詳しい説明は pycsw FAQ item を参照してください。FAQ 項目は pycsw 固有のものですが、一般的に他のカタログサーバーにも適用できます。

28.2.4 プラグインをオフラインで編集する

データ収集の場合、現場でノートパソコンや携帯電話がオフラインで作業することはよくある状況です。ネットワークに戻ったら、変更内容をマスターデータソース（PostGIS データベースなど）と同期させる必要があります。複数の人が同じデータセットで同時に作業している場合、同じ地物を変更していなくても、手作業で編集をマージするのは難しいものです。

 オフライン編集 プラグインは、データソースの内容を SpatiaLite または GeoPackage データベースにコピーし、オフライン編集を専用のテーブルに保存することで、同期を自動化します。再びネットワークに接続した後、オフライン編集をマスターデータセットに適用することができます。

プラグインを使うには

1. （Esri シェープファイル、PostGIS または WFS-T データソースなどから）いくつかのベクタレイヤを持ったプロジェクトを開きます
2. プラグインを有効にしている場合（:ref:core_and_external_plugins` を参照）:menuselection:**データベース --> オフライン編集 -->** **オフラインプロジェクトに変換** に進みます。その名前を冠したダイアログが開きます。
3. ストレージ型を選択します。*GeoPackage* または *SpatiaLite* データベース型を選択します
4. **ブラウズ** ボタンを使って、**オフラインデータ** を保存するデータベースの場所を指定します。既存のファイルでも作成するのでもかまいません。
5. **リモートレイヤを選択** セクションで、保存したいレイヤにチェックを入れます。そのレイヤの内容はデータベースのテーブルに保存されます。

   注釈: ターゲットのデータベース形式はネイティブでリストをサポートしないので、オフライン編集プラグインは、{文字列、数値}リストフィールドをカンマ区切りの文字列フィールドに変換します。これにより、オフライン時にこれらのフィールドの内容を読み込んで編集することができます。

   元のレイヤとオフラインのレイヤの両方のフィールドを扱いたい場合は、*try()* と *array* 式関数を利用します。例:

   ```
   try(array_contains("field",1),array_contains(string_to_array("field"),1))
   ```

   ---

6. 選択範囲がある場合、**選択がある場合に選択地物だけを同期する** チェックボックスをオンにすることで、サブセットのみを保存して作業することができます。大きなレイヤの場合、これは非常に有効です。

   これで全部です！

7. プロジェクトを保存して現場に持って行きます。
8. オフラインでレイヤを編集します。
9. 再接続後、**データベース　オフライン編集**　 **同期** を使って変更をアップロードします。

注釈: オフラインで使用するレイヤは、**レイヤ** パネルに アイコンが表示されます。

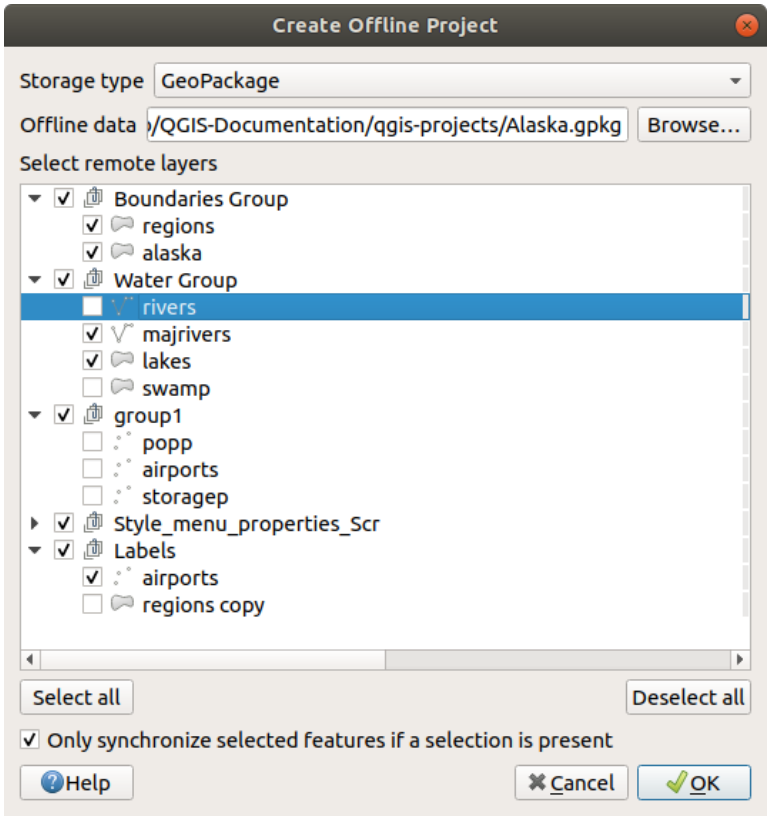


図 28.13: オフラインプロジェクトを作る

28.2.5 トポロジチェッカープラグイン

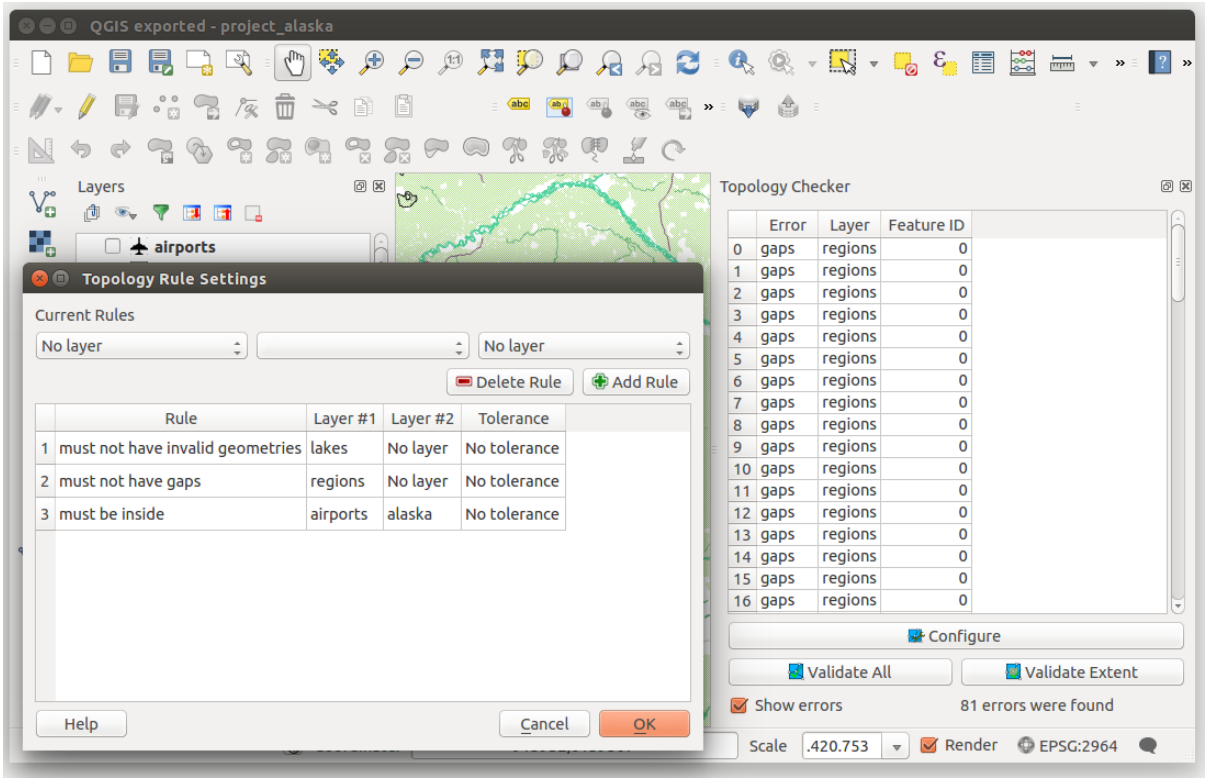


図 28.14: トポロジチェッカープラグイン

トポロジとは、地理的領域の地物を表すポイント、ライン、およびポリゴン間の関係を表すものです。トポロジチェッカープラグインを使用すると、ベクタファイルを調べて、いくつかのトポロジルールでトポロジを確認することができます。これらのルールは、地物の空間関係がお互いに「等しい」か、「含んでいる」か、「覆っている」か、「覆われている」か、「交わっている」か、「離れている」か、「交差している」か、「重複している」か、「接触している」か、または「範囲内にある」かどうかをチェックします。ベクタデータにどのトポロジルールを適用するかは、個別の質問によります。(たとえば、ラインレイヤのはみ出しは通常は容認しないでしょうが、これが道路の行き止まりを表している場合には、ベクタレイヤからそれらを削除することはしないでしょう)。

QGIS には、新たな地物をエラーなしで作成するのに最適な、組み込みのトポロジ編集機能があります。しかし、既存のデータエラーやユーザに起因するエラーを見つけるのは困難です。このプラグインは、ルールのリストを通じてこのようなエラーを見つけるのに役立ちます。

トポロジチェッカープラグインを使ってトポロジルールを作成するのは非常に簡単です。

ポイントレイヤ に対しては、次のルールが利用可能です:

- **次の地物に重ならなければならない**: プロジェクトのベクタレイヤ 1 つを選択できます。そのベクタレイヤのセグメント上にないポイントは「エラー」フィールドに挙げられます。
- **次の線の地物の両端に重ならなければならない**: プロジェクトのラインレイヤ 1 つを選択できます。ポイントはそのラインの端点と一致している必要があります。
- **次のポリゴンの内側でなくてはならない**: プロジェクトのポリゴンレイヤ 1 つを選択できます。ポ

イントはそのポリゴンの内側にある必要があります。そうでない場合には、QGIS はそのポイントを「エラー」に挙げます。

- **重複ジオメトリの地物があってはならない**: 同じ位置にポイントが 2 つ以上表示される場合には、「エラー」フィールドに挙げられます。
- **不正なジオメトリがあってはならない**: ジオメトリが有効であるかどうかをチェックします。
- **マルチパートがあってはならない**: マルチパートポイントは全て「エラー」フィールドに挙げられます。

ラインレイヤ に対しては、次のルールが利用可能です:

- **両端は次の点地物に重ならなければならない**: プロジェクトのポイントレイヤ 1 つを選択できます。ラインの両端はそのポイントと一致している必要があります。
- **ダングル（懸垂線）があってはならない**: ラインレイヤ内の線のはみ出しを表示します。
- **重複ジオメトリの地物があってはならない**: 同じ位置にラインが 2 つ以上重なって表示される場合には、「エラー」フィールドに挙げます。
- **不正なジオメトリがあってはならない**: ジオメトリが有効であるかどうかをチェックします。
- **マルチパートがあってはならない**: あるジオメトリが、実際には単純な（シングルパート）ジオメトリの集合となっていることがあります。このようなジオメトリはマルチパートジオメトリと呼ばれています。単純なジオメトリが 1 種類だけ含まれている場合には、これをマルチポイント、マルチラインまたはマルチポリゴンと呼んでいます。マルチパートのラインは全て「エラー」フィールドに挙げられます。
- **（2 つの地物を接続するだけの）疑似ノードがあってはならない**: あるラインジオメトリの端点は、他の 2 つのジオメトリと接続していなければなりません。他のジオメトリ 1 つだけと接続している場合には、この端点は疑似ノードと呼ばれます。

ポリゴンレイヤ に対しては、次のルールが利用可能です:

- **次の地物の少なくとも 1 点を包含しなければならない**: ポリゴンレイヤは 2 番目のレイヤの点ジオメトリの少なくとも 1 つを内部に含んでいる必要があります。
- **重複ジオメトリの地物があってはならない**: 同じレイヤのポリゴンは同一のジオメトリがあってはいけません。同じ位置にポリゴン地物が 2 つ以上重なって表示される場合には、「エラー」フィールドに挙げられます。
- **隙間（gap）があってはならない**: 隣接するポリゴンの間で隙間（gap）を形成してはいけません。行政区画が例として挙げられます（米国の州のポリゴンの間には隙間はありません…）。
- **不正なジオメトリがあってはならない**: ジオメトリが有効であるかどうかをチェックします。有効なジオメトリを定義するルールの中には次のようなものがあります。
  - ポリゴンのリングは閉じている必要があります。
  - 穴を定義するリングは、外側の境界を定義するリングの内側にある必要があります。
  - リングは自己交差してはいけません（互いに接触することも交差することもできません）。
  - リングは、1 点で接触する場合を除き、他のリングに接触できません。

- **マルチパートがあってはならない**: あるジオメトリが、実際には単純な（シングルパート）ジオメトリの集合となっていることがあります。このようなジオメトリはマルチパートジオメトリと呼ばれています。単純なジオメトリが 1 種類だけ含まれている場合には、これをマルチポイント、マルチラインまたはマルチポリゴンと呼んでいます。例えば、複数の島からなる国は、マルチポリゴンとして表現できます。
- **重なりがあってはならない**: 隣接するポリゴンは共通する領域を持ってはいけません。
- **次のポリゴン地物と重なりがあってはならない**: あるレイヤのポリゴンは、もう 1 つのレイヤの隣接するポリゴンと共通する領域を持ってはいけません。

以下は、QGIS で提供されるコアプラグインのリストです。これらは必ずしもデフォルトで有効になっているわけではありません。

アイコン	プラグイン	説明	マニュアルの参照
	DB マネージャ	QGIS からデータベースを管理する	*DB* **マネージャプラグイン**
	ジオメトリチェッカー	ベクタジオメトリのエラーの確認と修正	**ジオメトリチェッカープラグイン**
	GRASS 7	GRASS functionality	*GRASS GIS* **の統合**
	GRASS GIS プロバイダ	GRASS GIS プロセシング機能	*GRASS GIS* **の統合**
	MetaSearch Catalog Client	メタデータカタログサービス (CSW) との対話処理	*MetaSearch Catalog Client*
	オフライン編集	オフライン編集とデータベースとの同期	**プラグインをオフラインで編集する**
	OrfeoToolbox プロバイダ	OrfeoToolbox プロセシングプロバイダ	*OTB* **アプリケーションプロバイダ**
	プロセシング	空間データプロセシングフレームワーク	*QGIS* **プロセシングフレームワーク**
	SAGA GIS プロバイダ	SAGA GIS プロセシングプロバイダ	*SAGA: System for Automated Geoscientific Analyses*、**地球科学自動分析システム**
	トポロジチェッカー	ベクタレイヤのトポロジエラーを見つける	**トポロジチェッカープラグイン**

注釈: コア・プラグイン、  GRASS 7、  GRASS GIS プロバイダ、  OrfeoToolbox プロバイダ又は  SAGA GIS プロバイダを使うには、それらが構成されていなければなりません。 詳細は **ここ** にあります。

28.3 QGIS Python コンソール

この章の後半でご覧になるように、QGIS はプラグインアーキテクチャで設計されています。プラグインは Python で書くことができます。Python は地理空間分野では非常に有名なプログラミング言語です。

QGIS には、ユーザがオブジェクト（レイヤ、地物、インターフェイス）を対話的に利用できるように、Python API があります（コードのサンプルは PyQGIS 開発者用クックブック を参照して下さい）。また QGIS には Python コンソールもあります。

QGIS Python コンソールは Python コマンド実行用の対話型シェルです。自分の Python スクリプトを編集して保存するための Python ファイルエディタも付属しています。コンソールとエディタはどちらも PyQScintilla2 パッケージに基づいています。コンソールを開くには **プラグイン** *Python* **コンソール** (`Ctrl+Alt+P`) を選択します。

28.3.1 対話型コンソール

対話型コンソールは、ツールバー、入力エリア、および出力エリアから構成されています。

ツールバー

ツールバーからは以下のツールを利用することができます。

-  コンソールのクリア ：出力エリアをクリアします。
-  コマンドの実行 ：入力エリアでのみ使用可能です。 `Enter` キーを押すのと同じです。
-  エディタの表示 ： **コードエディタ** の表示・非表示を切り替えます。
-  オプション... ： コンソールのプロパティを設定するためのダイアログを開きます（ *Python* **コンソールの設定** を参照してください）。
-  ヘルプ... ：このドキュメントを閲覧します。

コンソール

コンソールの主な機能は以下のとおりです。

- コード補完、シンタックスハイライト、加えて以下の API には入力候補をポップアップで表示します。
  - Python
  - PyQGIS
  - PyQt5
  - QScintilla2
  - osgeo-gdal-ogr

- Ctrl キー+Alt キー+Space ： *Python* **コンソールの設定** で有効にしている場合は自動補完リストを表示する；
- キーボード入力し Enter または **コマンドを実行** を押して入力領域からコードスニペットを実行します。
- コンテキストメニューから **選択部分を入力** を使用するか、 Ctrl キー+ E を押して、出力領域からコードスニペットを実行します。
- Up と Down の矢印キーを使用して、入力エリアからコマンド履歴を閲覧して、必要なコマンドを実行します。
- Ctrl キー+シフト+ Space はコマンド履歴を表示するために：行をダブルクリックしてコマンドを実行します。 **コマンド履歴** ダイアログはまた、入力エリアのコンテキストメニューからアクセスできます。
- コマンドの履歴を保存して消去します。履歴はアクティブな *user profile* フォルダの下の console_history.txt ファイルに保存されます；
- QGIS C++ API のドキュメントを _api と入力して開きます；
- QGIS Python API のドキュメントを _pyqgis と入力して開きます。
- PyQGIS Cookbook を _cookbook と入力して開きます。

Tip: 出力パネルから実行済コマンドを再利用します

Ctrl キー+ E いくつかのテキストを選択し、を押すと出力パネルからコードスニペットを実行できます。選択したテキストは、インタープリタプロンプト (>>>, ...) が含　まれているかどうかは関係ありません。

図 28.15: Python コンソール

28.3.2 コードエディタ

エディタウィジェットを有効にするには  Show Editor ボタンを使います。Python ファイルの編集と保存が可能で、コードを管理するための高度な機能を提供します (コードのコメントとアンコメント、文法のチェック、GitHub 経由でのコードの共有など)。主な機能は以下の通りです：

- コード補完、シンタックスハイライト、加えて以下の API には入力候補をポップアップで表示します。
  - Python
  - PyQGIS
  - PyQt5
  - QScintilla2
  - osgeo-gdal-ogr
- `Ctrl` キー+`Space` で自動補完のリストを表示。
- コードスニペットを *GitHub* 経由で共有する。
- `Ctrl`+`4` 構文チェック。
- 検索バー（デフォルトのデスクトップ環境のショートカット、通常は `Ctrl`+`F` で開きます）：
  - 次/前を検索するにはデフォルトのデスクトップ環境のショートカットを使用します（`Ctrl` キー+`G` と `Shift` キー+`Ctrl` キー+`G`）。
  - 検索ボックスに入力すると自動的に最初の一致を見つけます。
  - 検索を開くと選択範囲に最初の検索文字列を設定します。
  - `Esc` を押すと検索バーを閉じます。
- オブジェクトインスペクタ：クラスと機能ブラウザ。
- （オブジェクトインスペクタから）マウスでクリックすると、オブジェクトの定義に移動します。
- コンテキストメニューの  *Run Selected* コマンドでコードスニペットを実行します；
- スクリプト全体を  *Run Script* コマンドで実行します（これにより、拡張子 `.pyc` のバイトコンパイルされたファイルが作成されます)。

注釈: **コードエディタ** は部分的または完全にからスクリプトを実行すると、コンソール出力領域に結果を出力します。

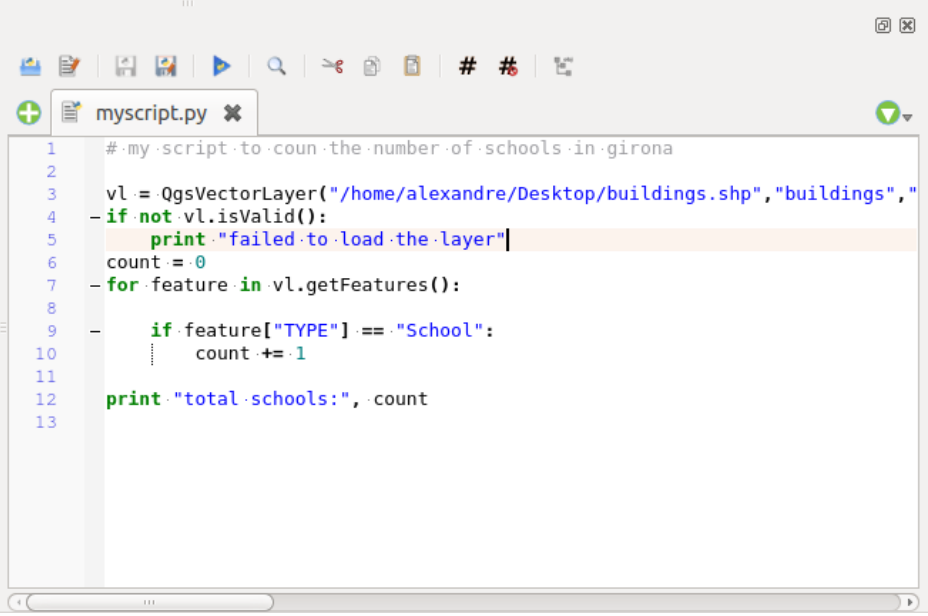


図 28.16: Python コンソールエディタ

Tip: オプションを保存

コンソールのウィジェットの状態を保存するには、[閉じる] ボタンから Python コンソールを閉じる必要があります。これにより、次の開始時に復元するためのジオメトリを保存できます。

第29章　ヘルプとサポート

29.1 メーリングリスト

QGIS は活発に開発中であり、常に期待のとおりに動作するとは限りません。助けを得るための好ましい方法は、qgis-users のメーリングリストに参加することです。あなたの質問は、より多くの人に届き、回答は他の人の利益にもなります。

29.1.1 QGIS ユーザー

このメーリングリストは、QGIS についての一般的な議論のためだけでなく、QGIS のインストールや使用に関する具体的な質問にも使用されています。次の URL にアクセスすると、qgis-users メーリングリストを購読できます： https://lists.osgeo.org/mailman/listinfo/qgis-user

29.1.2 QGIS 開発者

もしあなたが開発者で、より技術的な性質の問題に直面しているならば、qgis-developer メーリングリストに参加するのが良いでしょう。このメーリングリストは、QGIS に関連する UX（ユーザー・エクスペリエンス）やユーザビリティの問題を収集し、議論する場でもあります。以下にアクセスしてください： https://lists.osgeo.org/mailman/listinfo/qgis-developer

29.1.3 QGIS コミュニティチーム

このメーリングリストでは、ドキュメント、状況に応じたヘルプ、ユーザーガイド、ウェブサイト、ブログ、メーリングリスト、フォーラム、および翻訳作業といった話題を扱っています。ユーザーガイドで作業したい場合にも、このメーリングリストは質問をする良い出発点です。このメーリングリストは以下で購読できます： https://lists.osgeo.org/mailman/listinfo/qgis-community-team

29.1.4 QGIS 翻訳者

このメーリングリストでは、翻訳作業に関する話題を扱っています。ウェブサイトやマニュアル、インターフェース（GUI）の翻訳に携わりたいと考えているならば、このメーリングリストは質問をする良い出発点です。このメーリングリストは以下で購読できます： https://lists.osgeo.org/mailman/listinfo/qgis-tr

29.1.5 QGIS プロジェクト運営委員会（PSC）

このメーリングリストは、QGIS の全体的なマネジメントと方向性に関連する運営委員会（Steering Committee）の問題を議論するために使用されています。このメーリングリストは以下で購読できます：https://lists.osgeo.org/mailman/listinfo/qgis-psc

29.1.6 QGIS ユーザーグループ

QGIS を各国でプロモーションし、その開発に貢献するため、一部の QGIS コミュニティは QGIS ユーザーグループに組織されています。これらのグループは、地域のトピックを議論したり、地域または国内のユーザーミーティングを開催したり、機能開発のスポンサーを組織したりする場となっています。現在のユーザーグループの一覧は、 https://qgis.org/en/site/forusers/usergroups.html で確認できます。

どのメーリングリストへの登録も大歓迎です。質問に回答したり、経験を共有したりして、メーリングリストに貢献することを忘れないでください。

29.2 Matrix / IRC

Matrix (https://matrix.org) は分散型チャットのプロジェクトです。QGIS には#qgis:osgeo.org というエイリアスでアクセスできるルームがあり、libera.chat #qgis IRC チャンネルに橋渡ししています。

Matrix を使うには:

1. アカウントを作成します（matrix.org が最も簡単ですが、OSGeo アカウントをお持ちの場合は、OSGeo ID を matrix ID として使用することもできます）
2. クライアントをインストールします（Elements が最も簡単ですが、詳しくは `Matrix クライアント <https://matrix.org/docs/projects/try-matrix-now/#clients>`_を参照してください）か、またはブラウザで https://matrix.to/#/#qgis:osgeo.org 。

IRC を使うには:

1. IRC クライアントをインストールする
2. irc://irc.libera.chat/#qgis に接続するか、ブラウザで https://web.libera.chat/?channels=#qgis にアクセスします

29.3 商用サポート

QGIS の商用サポートも利用可能です。詳細情報については、 https://qgis.org/en/site/forusers/commercial_support.html のウェブサイトを確認してください。

29.4 バグトラッカー

qgis-users メーリングリストは、一般的な「QGIS で〜するにはどうしたらよいでしょうか？」形式の質問をするには便利ですが、QGIS のバグについて知らせたいときもあるでしょう。その時には、QGIS bug tracker を使用してバグレポートを提出できます。

あなたのバグは、必ずしもあなたが希望する優先順位を得られない可能性があることを心に留めておいてください（そのバグの重大性によります）。バグによっては、修正するために開発者が多大な労力を必要とする場合があり、いつでもそのために人手が割けるわけではありません。

機能のリクエストは、バグの場合と同じチケットシステムを使用して提出することができます。その際には、ラベルの種類に必ず `Feature request` を選択してください。

バグを発見し、自分で修正した場合には、 Github QGIS Project でプルリクエストを送信してください。

詳細については、 バグ、機能、問題 および submit_patch を参照してください。

29.5 Blog

また、QGIS コミュニティは https://plugins.qgis.org/planet/ でブログを運営しており、ユーザーや開発者にとって興味深い記事があります。他にも多数の QGIS ブログが存在しています。あなたも自分の QGIS ブログで貢献してみませんか？

29.6 プラグイン

https://plugins.qgis.org のウェブサイトは、公式の QGIS プラグインポータルです。ここには、「QGIS オフィシャルプラグインリポジトリ」経由で利用可能なすべての安定版および実験版 QGIS プラグインのリストがあります。

29.7 Wiki

最後に、私たちは https://github.com/qgis/QGIS/wiki で WIKI ウェブサイトを管理しています。WIKI には QGIS の開発に関する情報やリリース計画、ダウンロードサイトへのリンク、 メッセージ交換のヒントなど、さまざまな有用な情報があります。ぜひご覧ください。

第30章　協力者

QGIS は献身的なボランティアや組織のチームによって開発されたオープンソースプロジェクトです。私たちは人種、信条、性別、その人の生き方にかかわらず人々を歓迎するコミュニティになるよう努めています。あなたはいつでも私たちのコミュニティに 参加 できます。

30.1 著者

QGIS 全体のドキュメントについて、書き込み、レビュー、および更新に自分の時間とエネルギーを捧げる人たちを以下に記載します。

ティム・サットン	Yves Jacolin	Jacob Lanstorp	Gary E. Sherman	Richard Duivenvoorde
Tara Athan	Anita Graser	Arnaud Morvan	Gavin Macaulay	Luca Casagrande
K. Koy	Hugo Mercier	Akbar Gumbira	Marie Silvestre	Jürgen E. Fischer
Fran Raga	Eric Goddard	Martin Dobias	Diethard Jansen	Saber Razmjooei
Ko Nagase	Nyall Dawson	Matthias Kuhn	Andreas Neumann	Harrissou Sant-anna
Manel Clos	David Willis	Larissa Junek	Paul Blottière	Sebastian Dietrich
Chris Mayo	Stephan Holl	Magnus Homann	Bernhard Ströbl	Alessandro Pasotti
N. Horning	Radim Blazek	Joshua Arnott	Luca Manganelli	Marco Hugentobler
Andre Mano	Mie Winstrup	Frank Sokolic	Vincent Picavet	Jean-Roc Morreale
Andy Allan	Victor Olaya	Tyler Mitchell	René-Luc D'Hont	Marco Bernasocchi
Ilkka Rinne	Werner Macho	Chris Berkhout	Nicholas Duggan	Jonathan Willitts
David Adler	Lars Luthman	Brendan Morely	Raymond Nijssen	Carson J.Q. Farmer
Jaka Kranjc	Mezene Worku	Patrick Sunter	Steven Cordwell	Stefan Blumentrath
Andy Schmid	Vincent Mora	Alexandre Neto	Hien Tran-Quang	Alexandre Busquets
João Gaspar	Tom Kralidis	Alexander Bruy	Paolo Cavallini	Milo Van der Linden
Peter Ersts	Ujaval Gandhi	Dominic Keller	Giovanni Manghi	Maximilian Krambach
Anne Ghisla	Dick Groskamp	Uros Preloznik	Stéphane Brunner	QGIS Korean Translator
Zoltan Siki	Håvard Tveite	Matteo Ghetta	Salvatore Larosa	Konstantinos Nikolaou
Tom Chadwin	Larry Shaffer	Nathan Woodrow	Martina Savarese	Godofredo Contreras
Astrid Emde	Luigi Pirelli	Thomas Gratier	Giovanni Allegri	GiordanoPezzola
Paolo Corti	Tudor Bărăscu	Maning Sambale	Claudia A. Engel	嘉山 陽一
Otto Dassau	Denis Rouzaud	Nick Bearman	embelding	ajazepk
Ramon	Andrei	zstadler	icephale	Rosa Aguilar
Patrice Pineault	Jörn Gutzeit	Felix Feckler	Benoît de Mezzo	Étienne Trimaille
Andrea Giudiceandrea	Julien Cabieces	roya0045	Sebastian Gutwein	Jessica Veenstra
Ryan Welfle	Martin Pergler	Ivan Ivanov	muranamihdk	Loïc Bartoletti
Tomasz Taraś	Ian Maddaus	Jürnjakob Dugge	Roman Bug	Damiano Lombardi

次のページに続く

表 30.1 – 前のページからの続き

Marc Ducobu	Philip Albrecht	Dennis Milechin	Cody Martin	Savinaud Mickaël
Stefan Uhrig	Ariadni-Karolina Alexiou	Björn Hinkeldey	Benjamin Riley	MorriganR
Thayer Young	Shane Carey	Ian Turton	Emma Hain	Germán Carrillo
Jakob Miksch	Nicolas Boisteault	Bertrand Rix	Jorge Rosales	

30.2 翻訳者

QGIS は多言語アプリケーションであり、現状でいくつかの言語に翻訳されたドキュメントを公開しています。他の多くの言語でも翻訳が行われていて、一定の翻訳率を超えるとそれらは公開されます。あなたが翻訳の改善を支援したい、あるいは新しい言語の翻訳を要求したい場合は、https://qgis.org/en/site/getinvolved/index.html を参照してください。

現在の翻訳は以下の皆様のおかげで可能となっています：

言語	協力者
インドネシア語	Emir Hartato, I Made Anombawa, Januar V. Simarmata, Muhammad Iqnaul Haq Siregar, Trias Aditya
中国語（繁体字）	Calvin Ngei, Zhang Jun, Richard Xie
中国語（簡体字）	Xu Baocai
オランダ語	Carlo van Rijswijk, Dick Groskamp, Diethard Jansen, Raymond Nijssen, Richard Duivenvoorde, Willem Hoffman
フィンランド語	Matti Mäntynen, Kari Mikkonen
フランス語	Arnaud Morvan, Augustin Roche, Didier Vanden Berghe, Dofabien, Etienne Trimaille, Francis Gasc, Harrissou Sant-anna, Jean-Roc Morreale, Jérémy Garniaux, Loïc Buscoz, Lsam, Marc-André Saia, Marie Silvestre, Mathieu Bossaert, Mathieu Lattes, Mayeul Kauffmann, Médéric Ribreux, Mehdi Semchaoui, Michael Douchin, Nicolas Boisteault, Nicolas Rochard, Pascal Obstetar, Robin Prest, Rod Bera, Stéphane Henriod, Stéphane Possamai, sylther, Sylvain Badey, Sylvain Maillard, Vincent Picavet, Xavier Tardieu, Yann Leveille-Menez, yoda89
ガリシア語	Xan Vieiro
ドイツ語	Jürgen E. Fischer, Otto Dassau, Stephan Holl, Werner Macho
ヒンディー語	Harish Kumar Solanki
イタリア語	Alessandro Fanna, Anne Ghisla, Flavio Rigolon, Giuliano Curti, Luca Casagrande, Luca Delucchi, Marco Braida, Matteo Ghetta, Maurizio Napolitano, Michele Beneventi, Michele Ferretti, Roberto Angeletti, Paolo Cavallini, Stefano Campus
日本語	馬場 美彦、赤木 実、山手 規裕、水谷 貴行、縫村 崇行、嘉山 陽一
韓国語	OSGeo Korean Chapter
ポーランド語	Andrzej Świąder, Borys Jurgiel, Ewelina Krawczak, Jakub Bobrowski, Mateusz Łoskot, Michał Kułach, Michał Smoczyk, Milena Nowotarska, Radosław Pasiok, Robert Szczepanek, Tomasz Paul
ポルトガル語	Alexandre Neto, Duarte Carreira, Giovanni Manghi, João Gaspar, Joana Simões, Leandro Infantini, Nelson Silva, Pedro Palheiro, Pedro Pereira, Ricardo Sena
ポルトガル語（ブラジル）	Arthur Nanni, Felipe Sodré Barros, Leônidas Descovi Filho, Marcelo Soares Souza, Narcélio de Sá Pereira Filho, Sidney Schaberle Goveia
ルーマニア語	Alex Bădescu, Bogdan Pacurar, Georgiana Ioanovici, Lonut Losifescu-Enescu, Sorin Călinică, Tudor Bărăscu
ロシア語	Alexander Bruy, Artem Popov
スペイン語	Carlos Dávila, Diana Galindo, Edwin Amado, Gabriela Awad, Javier César Aldariz, Mayeul Kauffmann, Fran Raga
ウクライナ語	Alexander Bruy

30.3 翻訳に関する統計

QGIS 3.22 長期リリースの翻訳の努力の成果は以下の通りです。バージョンリリースまでに 5%に達した言語のみを掲載しています。

最終更新 *2023-12-26*

文字列数	翻訳先の言語数	全体の翻訳率
29206	**58**	**14.9%**

言語	翻訳率（%）	言語	翻訳率（%）	言語	翻訳率（%）
アルバニア語	1.11	アラビア語	4.63	アゼルバイジャン語	0.87
バスク語	1.89	ベンガル語	1.05	ブルガリア語	3.41
ビルマ語	0.97	カタルーニャ語	2.03	中国語（簡体字）	29.71
中国語（繁体字）	2.19	クロアチア語	0.98	チェコ語	6.73
デンマーク語	1.58	オランダ語	100.0	エストニア語	2.24
フィンランド語	2.22	フランス語	85.34	ガリシア語	1.35
ジョージア語	0.98	ドイツ語	22.97	ギリシャ語	1.24
ヘブライ語	1.68	ヒンディー語	1.19	ハンガリー語	19.16
イボ語	0.87	インドネシア語	3.64	イタリア語	100.0
日本語	99.93	カビール語	0.97	韓国語	86.8
リトアニア語	11.46	マケドニア語	0.99	マレー語	0.9
マラヤーラム語	0.97	マラーティー語	1.01	モンゴル語	0.98
ンコ語	2.52	ノルウェー語（ブークモール）	3.75	Panjabi (Punjabi)	0.0
ペルシア語	0.68	ポーランド語	2.74	ポルトガル語（ブラジル）	70.57
ポルトガル語（ポルトガル）	8.55	ルーマニア語	34.84	ロシア語	15.44
セルビア語	0.13	スロバキア語	2.43	スロベニア語	3.66
スペイン語	99.96	スウェーデン語	1.94	タガログ語	0.97
タミル語	1.41	テルグ語	0.88	タイ語	0.98
トルコ語	3.36	ウクライナ語	3.67	ウルドゥー語	0.86
ベトナム語	1.11				

第31章　付録

31.1 付録 A: GNU一般公衆ライセンス（General Public License）

Version 2, June 1991

はじめに

ほとんどのソフトウェアのライセンスは、それをユーザーが共有し変更する自由を奪うように設計されています。これとは対照的に、 GNU 一般公衆利用 LICENSE は、フリーソフトウェアを共有し変更する自由を保証することを意図しています - そのソフトウェアをすべてのユーザーが自由に使用できることを確かにする。この一般公衆利用許諾契約書は、フリーソフトウェア財団のソフトウェアのほとんどに、その使用を委ねたその他のすべてのプログラムに適用されます。（いくつかのフリーソフトウェア財団のソフトウェアは、代わりに GNU ライブラリ一般公衆利用許諾契約書によって覆われている。）これはあなたのプログラムにも適用できます。

私たちがフリーソフトウェアと言うときは、自由ではなく、価格のことを言っています。この一般公衆利用許諾書は、あなたがソースコードを受け取るか、またはあなたがそれをしたい場合は変更でき、それを得ることができることを、あなたは（あなたが希望する場合は、このサービスのためにと電荷）フリーソフトウェアのコピーを配布する自由を持っていることを確認するために設計されています新しいフリー・プログラムの一部をソフトウェアまたは使用します。あなたが知っていることを、あなたはこれらの事を行うことができます。

あなたの権利を保護するために、我々はあなたにこれらの権利を否定したり、権利を放棄するか聞いて誰を禁じる制限を加える必要があります。あなたはそれを変更した場合、これらの制限は、あなたがソフトウェアのコピーを配布する場合は、あなたのために一定の責任が生じることになります。

たとえば、あなたは、このようなプログラムの複製物を頒布する場合有料か無料に関わらず、受領者にあなたが持っているすべての権限を与える必要があります。あなたは、彼らもソースコードを受信または入手できることを確認する必要があります。彼らが自分たちの権利を知るよう、あなたは彼らにこれらの条項を示さなければなりません。

私たちは 2 つのステップであなたの権利を保護します。(1) 著作権ソフトウェア、および (2) あなたにソフトウェアをコピー、配布および/または変更する法的許可を与えるこのライセンスを提供する。

また、各作成者や私たち自身を守るために、私たちは、誰もがこのフリーソフトウェアには何の保証がないことを理解していることを確かにしたいです。ソフトウェアが他の誰かによって変更され回覧された場

合、他の誰かによって持ち込まれたいかなる問題も原作者の名声に反映されないよう、私たちはその受取人に自分たちが持っているものがオリジナルではないことを知って頂きたいです。

最後に、どのフリープログラムもソフトウェア特許に絶えず脅かされています。私たちは、プログラムの独占所有権が効力を発揮して、フリープログラムの再頒布者が個別に特許ライセンスを取得する危険を避けたいと思います。これを防ぐために、私たちはどの特許がすべての人の自由な使用のためにライセンスを取得しているか、全くライセンスを取得していないしなければならないということを明確にしました。

複製、頒布、改変に対する正確な条項と条件を次に。複製や頒布、改変のための条項および条件

0. このライセンスは、この一般公衆利用許諾契約書の条件の下で配布できることを言う、著作権者により配置された通知を含むすべてのプログラムまたはその他の著作物に適用されます。「プログラム」、以下、そのようなプログラムや仕事を指し、「プログラムに基づいた作品は、」プログラムや著作権法の下で任意の派生物のいずれかを意味しますと言うことです、プログラムまたは一部を含む作品をそれ、どちらかそのままあるいは改変し、および/または他の言語に翻訳。（以下、翻訳は、用語「改変」で制限なく含まれています。）各ライセンシーは「あなた」として扱われます。

   複製、頒布、改変以外の活動はこのライセンスで保護されません。彼らはその範囲外です。プログラムを実行する行為は制限されず、プログラムからの出力はそのコンテンツがプログラムに基づく著作物（プログラムを実行することによってなされたとは独立）を構成している場合にのみ含まれます。それが本当かどうかは、当該プログラムに依存します。

1. あなたは受け取ったプログラムのソースコードの逐語的なコピーをコピーして配布することができ、任意の媒体に、あなたが顕著と適切に保証の適切な著作権表示と免責条項をコピーし、それぞれに掲載し、このライセンスと保証不在を参照するすべての通知を残し、そして、プログラムと一緒にプログラムの任意の他の受信者に本ライセンスのコピーを与えます。

   あなたは複製を譲渡する実際の行為に対して料金を請求でき、報酬と引き換えに保護の保証を提供してもよいです。

2. あなたはまた、これらの条件をすべて満たしていることを提供し、そのプログラムに基づいて仕事を形成し、あなたのコピーやプログラムのコピーまたはその一部を変更し、コピーして、上記第 1 の条件の下でそのような改変や仕事を配布することができます：

   a) あなたは、ファイルや変更の日付を変更する旨の著名な通知を運ぶために変更されたファイルを起こさなければなりません。

   b) あなたは、本ライセンスの条項の下ですべての第三者へ無償で全体として認可されるように、全体的にまたは部分的に含まれているか、プログラムまたはその任意の部分に由来していることを、あなたが配布するすべての作業が発生したり、公開する必要があります。

   c) 実行時に修正プログラムが正常に対話的にコマンドを読む場合は、印刷したり、適切な著作権表示および無保証がない旨の通知を含む告知を表示するために、最も一般的な方法で対話的に実行し始めたとき、それを引き起こす（または必要があります他に、あなたが保証を提供することを言う）と、ユーザーがこれらの条件の下でプログラムを再配布し、本ライセンスのコピーを表示する方法をユーザーに伝えるかもしれません。（例外は：プログラムそのものは対話的であっても、通常は、そのような声明を印刷しない場合は、プログラムに基づいて、あなたの仕事は、告知を印刷する必要はありません。）

   これらの要件は、全体として修正作業に適用されます。著作物の一部は、プログラムに由来しない、と合理的に独立した別の自分自身で作品は、このライセンス、およびその条件を考慮することができるならば、あなたが個別の著作物として頒布する場合、これらの部分には適用されません。あなたが

プログラムを基にした著作物全体の一部として、同じ部分を配布する場合でも、全体の分布は、パーミッション他のライセンシーのために全体の全体に及ぶ本ライセンスの条件になるため、それぞれに必要がありますとにかかわらず、それを書いた人のすべての部分。

したがって、権利を主張したり、あなたによって完全に記述された動作するようにあなたの権利に異議を申し立てることはこの節の意図するところではありません。むしろ、その意図は、プログラムに基づいた派生物や集合著作物の頒布を管理する権利を行使することです。

また、記憶装置または配布媒体のボリューム上のプログラム（またはプログラムに基づく著作物）を用いてプログラムに基づいていない他の作業の単なる集合は、このライセンスの範囲の下で他の作業をもたらしません。

3. あなたはまた、次のいずれかを実行すれば、セクション 1 と 2 上記の条件の下で、オブジェクトコードまたは実行可能形式でプログラム（またはそれに基づく仕事、第 2 節の下）をコピーして配布できます：

   a) 完全な対応する機械読み取り可能なソースコードをそれに付随します、これはソフトウェアの交換で習慣的に使用される媒体に、上記のセクション 1 および 2 の条件の下で配布されなければならない。または、

   b) 物理的に下で配布されるソースの配布、対応するソースコードの完全な機械読み取り可能なコピーを、実行するあなたの費用を超えない料金で、第三者を与えるために、少なくとも 3 年間有効、書かれた第三者に対しても、ソフトウェアの交換で習慣的に使用される媒体上で上記のセクション 1 および 2 の観点。または、

   c) あなたが対応するソースコード頒布の申し出に得た情報を一緒に引き渡すこと。（この代替は、非商用配布のために許可され、あなたが上記サブセクション B と一致して、このようなオファーとオブジェクトコードまたは実行可能形式のプログラムを受信した場合のみ）。

   作業のソースコードは、それに変更を行うための作業の好ましい形態を意味します。実行可能な作業のために、完全なソースコードは、それに含まれるすべてのモジュールに加え、関連するすべてのインターフェイス定義ファイル、および実行可能ファイルのコンパイルとインストールを制御するために使用するスクリプトのすべてのソースコードを意味しています。しかし、特別な例外として、ソースコードは、オペレーティングシステムの主要なコンポーネント（コンパイラ、カーネルなど）と共に（ソースまたはバイナリ形式のいずれかで）正常に分布しているものを含む必要はない分散その上で実行可能な実行、そのコンポーネント自体が実行可能を伴う場合を除きます。

   実行形式またはオブジェクトコードの頒布が、指定された場所からコピーするためのアクセスを提供することによって行われている場合は、第三者がコピーするように強要されていない場合でも、ソースコードの同じ場所からソースコードをコピーするために同等のアクセスを提供オブジェクトコードと一緒にソース。

4. あなたは、コピー、改変、サブライセンス、または明示本ライセンスの下で提供以外のプログラムを配布することはできません。そうでない場合は、コピー、改変、サブライセンスまたはプログラムを配布しようとする試みは無効となり、かつ自動的に本ライセンスの下であなたの権利を終了します。しかし、この契約書の下であなたから複製や権利を受け取った当事者は、そのライセンスがあれば、そのような当事者が完全に従って残るように終了していません。

5. あなたはそれに署名していないので、あなたは、このライセンスを受け入れる必要はありません。しかし、他には何もプログラムまたはその派生物を変更または頒布する許可を与えるものは存在しません。あなたが本ライセンスに同意しない場合、これらの行為は法律で禁止されています。したがって、プログラム（またはプログラムに基づく著作物）を変更または配布することによって、あなたは

それに基づいてプログラムまたは作品を配布したり、変更、コピーのためにそうするように、この契約書を受諾したということ、そしてそのすべての契約条件。

6. あなたがプログラム（またはプログラムに基づく著作物）を再配布するたびに、受信者は、自動的にこれらの条件にプログラムの対象を、複製、頒布または変更するオリジナルのライセンサーからライセンスを受けました。あなたはここで認められた権利の受信者の行使について、さらに制約を加えることはできません。あなたは、第三者がこの契約書に従うことを強制する責任はありません。

7. 、裁判所の判決または特許侵害のあるいはその他の理由（特許関係に限らない）のための申し立ての結果として、条件は本ライセンスの条件と矛盾する（裁判所の命令、契約またはそれ以外で）あなたに課せられている場合は、あなたがこの契約書の条件を免除されるわけではありません。同時にあなたの本ライセンスの下での義務およびその他の関連する義務を満たすよう、そう頒布できない場合は、すべてのプログラムを配布できません。例えば、あなたから直接間接を問わずコピーを受け取った人誰もがプログラムを使用料無料で再頒布することを特許ライセンスが認めていない場合、あなたがそのライセンスと本ライセンスの両方を満たすことができる唯一の方法は、プログラムの頒布を完全にやめることです。

   本節の任意の部分を任意の特定の状況下で無効または執行不能とされた場合、セクションのバランスが適用されることを意図されており、全体として部分は、他の状況に適用されることが意図されます。

   あなたが任意の特許やその他の財産権の主張を侵害したり、そのような主張の有効性を争うために誘導するために、このセクションの目的ではありません。このセクションはパブリックライセンス履行により実施されるフリーソフトウェア配布システムの完全性を保護する目的を持っています。多くの人々が、このシステムの一貫した適用を信頼して、このシステムを通じて配布されたソフトウェアの広い範囲への寛大な貢献をしてきました。それは彼または彼女がどのようなシステムを通じてソフトウェアを配布すると、ライセンシーはその選択を強要することはできません喜んでいるかどうかを判断するために、著者/ドナー次第です。

   このセクションでは、本ライセンスの残りの結果であると考えられるケースを徹底的に明らかにすることを目的としています。

8. プログラムの配布および/または使用が特許または著作権のあるインターフェイスにより、特定の国に制限されている場合は、その分布が許可されるように、本ライセンスの下でプログラムを元の著作権者は、これらの国を除く明示的な地理的頒布制限を加え、または除外されていない国の中だけ。このような場合には、このライセンスは、本ライセンスの本文に書かれたかのように制約を組み込みます。

9. フリーソフトウェア財団は、随時改訂および/または一般公衆利用許諾契約書の新バージョン発表することができます。このような新しいバージョンは、現在のバージョンとその精神においては似ていますが、新たな問題や懸念に対処するために細部では異なることがあります。

   各バージョンは、バージョン番号によって区別を与えています。プログラムはそれと「それ以降のバージョン」に適用される本ライセンスのバージョン番号を指定した場合は、そのバージョンのか、フリーソフトウェア財団によって発行されたそれ以降のバージョンのいずれかの条件を次のオプションがあります。プログラムが本ライセンスのバージョン番号を指定しない場合、あなたは今までにフリーソフトウェア財団によって発行されたバージョンを選択することができます。

10. プログラムの一部を頒布条件が異なる他のフリープログラムに組み込みたい場合は、許可を求めるために著者にご一報ください。フリーソフトウェア財団が著作権を保有するソフトウェアについては、フリーソフトウェア財団にご一報ください。私たちは時々、このための例外を作ります。私たちの決定は、私たちのフリーソフトウェアのすべての派生物がフリーな状態に保たれること、そして一般的

にソフトウェアの共有と再利用を促進という 2 つの目標によって導かれます。

無保証

11. プログラムは無償で許可されるので、適用法により認められる範囲で、プログラムの何の保証も無いです。その他の方法で「そのまま」AND / OR 他の当事者はプログラムを提供著作権所有書面で明記する場合を除き、いかなる保証もない現状、あるいは暗示含むがこれらに限定されない、特定目的に対する適合性の黙示の保証。プログラムの品質および性能に関するすべてのリスクはお客様が負うものとしています。プログラムに欠陥があると判明した場合、あなたは必要なすべてのサービス、修理または補正の費用を負担するものと。

12. 適用法で要求されるまたは書面で同意がある場合を除きいかなる場合においても、任意の著作権者、または上記で許可されプログラムを修正および/または再頒布するその他の当事者は、任意の一般的な、特別、付随的、または含む、損害 FOR YOU TO 責任を負わないものとします EVEN、（データの損失、または不正確レンダリングされているデータ、またはあなたや第三者、または他のプログラムと一緒に動作するプログラムの障害によって被った損失を含む、ただしこれらに限定されない）プログラムを使用または使用不能から生じる損害当該保有者又はその他の当事者は、そのような損害の可能性について知らされている場合。

GPL のための **QGIS Qt の例外**

加えて、特例として QGIS 開発グループは有償無償に関わらず、また以下に示すものに限らず Qt ライブラリへのコードのリンクを許可します。具体的には Qt/Non-commercial Windows, Qt/Windows, Qt/X11, Qt/Mac, そして Qt/Embedded (また、Qt と同じライセンスを適用している Qt の修正版もこれに含まれます)。これら 2 つを含むリンクされた組み合わせの配布についても許可します。利用者は GNU General Public License に従い、Qt 以外に使われている全てのコードを尊重してください。仮にこのファイルを変更する場合、変更したファイルに対してもこの例外規定の拡張、適用を考えるかもしれませんが、その義務はありません。この例外規定の適用を望まない場合には、変更後のファイルからこの例外規定に関する文章を削除してください。

31.2 付録 B: GNU フリー文書利用許諾契約書

Version 1.3, 3 November 2008

著作権 2000、2001、2002、2007、2008 年 フリーソフトウェア財団

https://www.fsf.org/

誰もがこのライセンス文書の逐語的なコピーを複製および頒布することは許可されていますが、それを変更することはできません。

はじめに

このライセンスの目的は、マニュアル、教科書、または他の機能的で便利な文書を、自由という意味で「フリー」にすることです。商業的または非商業的、またはそれを修正することなく、誰もがそれをコピーして再配布するための事実上の自由を確保するために。第二に、このライセンスは、他の人によって行われた変更の責任とみなされていないながら、著者や出版社が自分の仕事のためにクレジットを取得する方法を保持します。

このライセンスは「コピーレフト」の一種です。すなわち、文書の派生物自体は同じ意味でフリーでなければなりません。これは、フリーソフトウェアのために設計されたコピーレフトライセンスである GNU 一般公衆利用許諾契約書を、補完します。

フリーのプログラムは、ソフトウェアが行うのと同じ自由を提供するマニュアルが付属していなければならない：フリーソフトウェアはフリーな文書が必要なので、私たちは、フリーソフトウェアのマニュアルのためにそれを使用するために、このライセンスを設計しています。しかし、このライセンスは、ソフトウェアのマニュアルに限定されるものではありません。それは関係なく、主題のか、それが印刷された本として出版されているかどうか、任意のテキストの仕事のために使用できます。私たちは、その目的の命令または参照された作品のため、主にこのライセンスをお勧めします。

1．適用性と定義

このライセンスは、著作権者がこのライセンスの条件に基づいて頒布できることを示す通知を含む、あらゆる媒体のあらゆるマニュアルまたはその他の作品に適用されます。そのような通知は、ここに記載された条件の下でその作品を使用するための、期間無制限の、全世界のロイヤルティフリーライセンスを付与します。以下で **文書** は、そのようなマニュアルまたは文書を指します。一般会員はすべてライセンシーであり、「**あなた**」と呼ばれます。著作権法に基づく許可が必要な方法で作品をコピー、変更、または配布する場合、あなたはライセンスを受け入れます。

文書の「 **修正版** 」は、ドキュメントまたはその一部、逐語的、または変更してコピー及び/又は他の言語に翻訳のいずれかを含む任意の作業を意味します。

「 **補遺部分** 」は、名前の付録か（または関連事項）文書の全体的な対象への文書の出版社や著者の関係で独占的に扱う文書のフロントマター部であり、それは、その全体的な対象内で直接落下する可能性がありますが何も含まれていません。（文書が部分的に数学の教科書である場合はこのように、補遺部分は、任意の数学を説明できないことがあります。）関係は主題または関連事項との歴史的な関連の問題であるか、対象とか、あるいはそれらに関する法的、商業、哲学的、倫理的、あるいは政治的な位置の問題である可能性があります。

「 **変更不可部分** 」とは、二次著作物セクションであって、「文書」がこのライセンスに基づいてリリースされたという通知中で、そのタイトルが「変更不可部分」のタイトルとして指定されているものです。部分が上記の「二次著作物」の定義に適合しない場合、部分を「変更不可部分」として指定することはできません。文書には変更不可部分が含まれない場合があります。文書が変更不可部分を識別しない場合は何もありません。

「 **カバーテキスト** 」とは、短い文章であって、文書がこのライセンスの下でリリースされていることを述べている通知中でフロントカバーテキストまたはバックカバーテキストとしてリストされているものです。フロントカバーテキストは最大 5 語、バックカバーテキストは最大で 25 語が許されています。

文書の「 **透明** 」コピーは機械読み取り可能なコピーを意味し、その仕様一般公衆に利用可能な形式で表され、それは一般的なテキストエディタでまっすぐ文書の改訂のためか（構成画像に適していますピクセル）汎用ペイントプログラムまたは（図面のために）、いくつかの広く利用可能な描画エディタ、それはフォーマッタをテキストまたはテキストフォーマッタへの入力に適した様々なフォーマットへの自動翻訳のために入力するのに適しています。そのマークアップ、またはマークアップの非存在下、さもなければ透明ファイル形式で作られたコピーは、読者が後続の変更を妨害または阻止するように配置された透明でありません。テキストの任意のかなりの量のために使用した場合の画像フォーマットは、透明ではありません。「透明」でないコピーは **不透明** と呼ばれます。

透明な複製に適した形式の例としては、マークアップなしのプレーン ASCII、Texinfo の入力形式、LaTeX 入力形式、一般に入手可能な DTD を使用した SGML または XML、および標準に準拠した単純 HTML、人

間の変更のために設計された PostScript または PDF があります。透明な形式の例には、PNG、XCF および JPG があります。不透明な形式には、商用ワードプロセッサで読み取り編集できる商用形式、DTD および／またはプロセシングツールが一般に利用可能でない SGML または XML、および機械生成 HTML、いくつかのワードプロセッサによって作成された出力目的のみの PostScript や PDF があります。

「**題扉**」とは、印刷された書籍、題扉自体、プラスを保持するために必要とされるような以下のページのために、読みやすく、材料は、このライセンスは、題扉に表示されている必要があります。以下のような任意の題扉を持っていないフォーマットの作品については、「題扉」には、テキストの本文の先頭に先行し、作品の題の最も顕著な外観に近いテキストを意味します。

「**パブリッシャー**」は、公衆への文書のコピーを配布する個人または団体を意味します。

「**XYZ という題**」セクションには、題を正確 XYZ であるか、XYZ を別の言語に翻訳し、テキストを次の括弧内に XYZ が含まれているいずれかの文書の名前のサブユニットを意味します。（ここで、XYZ は「**謝辞**」、「**献呈**」、「**裏書**」、または「**歴史**」などのような、下記の特定のセクション名を表します）。この定義によると「文書」を修正するときにセクションの **題を保持** することは、XYZ という題のセクションを残すことを意味します。

ドキュメントは、本ライセンスは、ドキュメントに適用されると述べている通知に次の保証の免責事項を含めることができます。これらの保証免責事項は、この契約書では、唯一の保証を放棄に関して参照により含まれると考えている。これらの保証の免責が持っていることを他の含意は無効であり、このライセンスの意味には影響を与えません。

2．逐語的に忠実な複製

このライセンス、著作権表示、および本ライセンスが文書に適用されると述べるライセンス通知がすべてのコピーに再現されている、かつ、本ライセンスのものに一切の他の条件を追加していないという条件が満たされる限り、文書は、商業的にも非商業的にも、任意の媒体にコピーして配布できます。作成あるいは頒布するコピーの閲覧または再コピーを妨げたり制御するための技術的手段を使用することはできません。ただし、コピーと引き換えに報酬を受け取ることはできます。十分に多い数のコピーを配布する場合は、セクション 3 の条件にも従わなければなりません。

また、上記と同じ条件の下で、コピーを貸与でき、コピーを公に表示できます。

3．大量の複製

100 以上の番号文書の印刷されたコピー（または一般的にカバーを印刷したメディアまたはコピー）を公開し、文書のライセンス通知がカバーテキストが必要な場合は、はっきりと読みやすく、コピーをこれらすべてのカバーテキスト（表紙にフロントカバーテキスト、および背面カバーにバックカバーテキスト）を持ち運びするカバー内に同封しなければなりません。どちらのカバーもはっきりと読みやすいこれらのコピーの出版社として識別する必要があります。フロントカバーは等しく顕著な可視題のすべての単語との完全なタイトルを提示しなければなりません。加えて、表紙に他の材料を加えてもよいです。それらは、文書の題を保持し、これらの条件を満たしている限り、カバーに限定変更とコピーは、他の点では逐語的なコピーとして扱うことができます。

どちらかのカバーに必要なテキストが読みやすく収まらないほど膨大な場合、最初のものを実際の表紙に（適当に収まるくらい多く）記載されている入れて、残りを隣接するページに続けるべきです。

100 以上の番号文書の不透明コピーを公開または配布する場合は、各不透明コピーにまたはでから、一般的なネットワーク - コンピュータ・ネットワーク上の場所を機械可読トランスペアレント各不透明コピーと一緒にコピー、または状態を含んでいなければならないのいずれかパブリックを使用すると、パブリック標準のネットワーク・プロトコル文書の完全な透明コピー、追加材料の自由を使用してダウンロードす

るためのアクセス権を持っています。後者のオプションを使用する場合は、数量に不透明コピーの配布を開始するとき、少なくとも 1 年の最後の時間の後に、配布されるまで、この透明コピーは定められた場所でこのようにアクセス可能なままであることを保証するために、合理的に慎重な手順を実行する必要があります公衆にその版の不透明なコピー（直接、または代理店や小売店を通じて）。

コピーの任意の多数を再配布する十分前に、文書の更新バージョンを提供する機会を与えるために、文書の作成者に連絡することは、要求されますが、必須ではありません。

4．変更

「文書」の「変更版」は、その「変更版」をまさにこのライセンスの下でリリースしていて、その「変更版」が「文書」の役割を満たしている、したがってそのコピーを所有している誰にでも「変更版」の配布、変更を許可している限り、上記のセクション 2 と 3 の条件の下でコピーおよび配布できます。また、「変更版」では以下のことを行う必要があります：

A. 題扉（とカバー、もしあれば）には文書の題、および以前の版（あった場合には文書の「履歴」セクションに表示されているはずです）の題とは異なる題を使用してください。その版の元の出版社が許可を与える場合は、以前の版と同じ題扉を使用できます。

B. 題扉のリストには、彼らはこの要件からあなたを解放しない限り、著者として、変更版における変更の著作者として責任がある 1 人以上の人または団体を、文書の主著者の少なくとも 5（5 より少ない場合その主著者のすべて）と一緒に、列挙します。

C. 題扉に修正版の出版社の名前を、出版社として、述べます。

D. 文書のすべての著作権表示を残します。

E. 他の著作権表示の近くに、あなたの修正のための適切な著作権表示を追加します。

F. 、すぐに著作権表示の後に、下記の補遺に示されている形で、本ライセンスの条項の下で変更版を使用する公開許可を与えるライセンス通知を含めます。

G. そのライセンスに保存するには、不変のセクションの完全なリストを気づくと文書のライセンス通知に与えられたカバーテキストを必要としていました。

H. 本ライセンスの変更されていないコピーが含まれます。

I. 「履歴」と題するセクションを保持し、その題を保持し、それに題ページに与えられたとして、修正版の、少なくとも題、年、新しい著者、および出版社を明記のアイテムを追加します。文書に「履歴」と題した章が存在しない場合は、その題ページに与えられたとして、文書の題、年、著者、および出版社を述べるものを作成し、その後、前の文で述べたように、変更版を記述する項目を追加します。

J. もしあれば、ネットワークの場所を保存する、パブリックアクセスの文書の透明コピーへ、およびそれが基づいていた以前のバージョンの文書に与えられ、同様に、ネットワークの場所のために文献で示さ。これらは、「履歴」セクションに配置できます。あなたは、4 年前に、文書自体、少なくとも出版された仕事のためのネットワークの場所を省略でき、またはそれが参照するバージョンのオリジナルの出版社は、許可を与える場合。

K. 「謝辞」または「献呈」と題された任意のセクションは、セクションの「題を保持」し、セクションの寄稿者の肯定応答および/またはその中に与えられた献呈の各々の全ての物質とトーンを維持します。

L. その本文および題名を変更せず、文書のすべての不変のセクションを保持します。章番号やそれに相当するものは、セクション題の一部とはみなされません。

M. 「裏書」と題されたいずれかのセクションを削除します。このようなセクションは、修正版には含まれないことがあります。

N. 「裏書」または任意の不変セクションとの題で競合する権利を有することに任意の既存のセクションを改称しないでください。

O. 任意の保証免責を保存します。

修正版は二次著作物セクションとしての資格や文書からコピーされた何の材料を含まない新しいフロントマターセクションまたは付録が含まれている場合、自身の選択によりこれらの一部または全部を不変として指定できます。これを行うには、変更版の利用許諾告知における変更不可部分のリストに自分の題を追加します。これらの題は、他のセクションの題は区別しなければなりません。

あなたは、「推薦の辞」と題されたセクションを追加します。例えば---それは、様々な関係者によってあなたの変更版の推薦しか含まれていない提供し、ピアレビューのステートメントまたはテキストは、標準の権威ある定義として組織によって承認されたことがあり。

変更版ではカバーテキストのリストの最後に、バックカバーテキストとして、最大 5 つのフロントカバーテキストなどの単語、および最大 25 ワードの通路の通過を追加できます。フロントカバーテキストとバックカバーテキストの 1 の唯一の通路はによって（またはによって行われた取り決めにより）いずれかのエンティティを添加してもよいです。文書が既に同じカバーするためのカバーテキストが含まれている場合は、以前にあなたによってか、の代わりに動作している同じエンティティによって行われた配置で追加された、別のものは追加できません。しかし、古い文を加えた以前の出版者からの明示的な許可に、古いものを置き換えることができます。

ドキュメントの作者と出版社（単数または複数）は、このライセンスによってのための宣伝のために自分の名前を使用するか、いずれかの修正版の裏書を主張または暗示する許可を与えることはありません。

5．書類を組み合わせる

無修正、変更版に関して上記のセクション 4 で定義された条件の下で、組み合わせて、オリジナルの文書のすべての不変のセクションのすべてが含まれていることを提供し、このライセンスの下で発表された複数の文書を結合し、それらをすべてリストアップしますその利用許諾告知にご組み合わせた作品のように不変のセクション、すべての彼らの保証の免責事項を保持しています。

結合後の著作本ライセンスのコピーが含まれているのみ必要とし、複数の同一の不変のセクションは、単一のコピーで置き換えることができます。同じ名前が異なる内容の変更不可部分が複数ある場合は、括弧内に、それの最後に追加することによって、そのような各セクション独特の題を作る、そのセクションの原作者や出版社の名前は、他の知られている場合、または一意の番号。結合後の著作物の利用許諾告知における変更不可部分の一覧で、章の題名に同様の調整を行います。

組み合わせでは、「履歴」と題する一つのセクションを形成し、様々なオリジナルの文書中の「履歴」という題のすべてのセクションを組み合わせなければなりません。「謝辞」という題のすべてのセクション、および「献呈」という題のすべてのセクションも同様に組み合わせます。「推薦」という題のすべてのセクションは削除する必要があります。

6．文書のコレクション

文書および本ライセンスの下でリリースされた他の文書からなるコレクションを作成し、コレクションに含まれる単一のコピーで様々な文書中のこのライセンスの個々のコピーを置き換えることは、他のすべての点で文書の各逐語的にコピーについてこのライセンスの規則に従う限りにおいて、許可されます。

このようなコレクションから単一の文書を抽出して個別に配布することは、このライセンスのコピーを抽

出された文書に挿入し、その文書の逐語的なコピーに関して他のすべての点で本ライセンスに従うかぎり、このライセンスの下で許可されます。

7．独立した作品でのまとめ

編集から生じた著作権は法律上の権利を制限するために使用されていない場合は、他の別個の独立した文書や作品で、またはストレージまたは配布媒体のボリューム上の文書またはその誘導体の編纂は、「まとめ」と呼ばれています、個々の作品は許可している以上、編纂のユーザーの。「文書」がまとめに含まれている場合、それ自体が「文書」の派生物ではないまとめ中の他の作品には、このライセンスは適用されません。

セクション 3 のカバーテキスト要件文書のこれらのコピーに適用可能である場合文献は、全体集合体の半分未満である場合、次に、文書のカバーテキストは、集合内の文書を囲むカバー上に配置されてもよい、またはカバーの電子同等の文書は、電子形式である場合。そうでなければ、彼らは全体の集計を一括印刷カバーの上に表示される必要があります。

8．翻訳

翻訳は変更の一種と考えられているので、翻訳で不変のセクションを交換部 4 の条件の下での文書の翻訳を配布することが彼らの著作権者からの特別な許可が必要ですが、に加えて、一部またはすべての不変のセクションの翻訳を含むことができこれらの不変のセクションの元版。また、本ライセンスの元の英語版およびそれらの通知および免責事項の元版が含まれていることを提供し、このライセンスの翻訳、および文書内のすべてのライセンス通知、および任意の保証免責事項を含むことができます。翻訳と本ライセンスまたは通知または免責条項の元版との間に食い違いが生じた場合は、元版が優先されます。

文書内のセクションは、「謝辞」、「献呈」と題された、または「履歴」である場合、要件（セクション 4）はその題（セクション 1）を維持するために、典型的には実際の題を変更する必要があります。

9．終了

コピー、変更、サブライセンス、または明示本ライセンスの下で提供以外の文書を配布することはできません。そうでない場合は、コピー、変更、サブライセンス、またはそれを配布しようとすると無効となり、かつ自動的に本ライセンスの下であなたの権利を終了します。

あなたがこの契約書のすべての違反をやめる場合は、その後、特定の著作権者からライセンスは、著作権者が明示的に、最終的には永久ライセンス、および（B）を終了しない限り、とまでは（a）の仮、著作権者が失敗した場合復活さ中止後 60 日前にいくつかの合理的な手段で違反を通知します。

著作権者は、いくつかの合理的な手段で違反を通知した場合また、特定の著作権者からライセンスを永続的に復活され、これはあなたがその著作権者から（すべての作業のための）本ライセンスの違反の通知を受けたのは初めてで、そしてあなたは、予告のあなたの受領後 30 日前に違反を治します。

このセクションの下であなたの権利の終了は、このライセンスの下であなたから複製や権利を受け取った当事者のライセンスは終了しません。あなたの権利が終了し、恒久的に回復いないされている場合は、同じ材料の一部または全部のコピーの受領はあなたにそれを使用する権利を与えるものではありません。

10．本ライセンスの将来の改訂

フリーソフトウェア財団は随時 GNU フリードキュメントライセンスを更新しています。新しいバージョンは既存のバージョンと同様の精神のもとにありますが、新しい課題や関心事について詳細では異なる見解を述べることもあります。 詳しくは http://www.gnu.org/copyleft/ を参照してください。

ライセンスの各バージョンは、バージョン番号によって区別を与えています。文書は、本ライセンスの特定の番号のバージョン「またはそれ以降のバージョンが」それに適用され、あなたがその指定されたバージョンのか、と（いない出版されている任意の以降のバージョンのいずれかの条件を次のオプションを持っ

ていることを指定した場合フリーソフトウェア財団によって草案)。文書が本ライセンスのバージョン番号が指定されていない場合は、フリーソフトウェア財団によってかつてない（ないドラフトとして）発行されたバージョンを選択することができます。文書がプロキシは、このライセンスの将来のバージョンを使用できるかを決定できるように指定した場合は、バージョンの受け入れのそのプロキシの公開声明は、恒久的に文書のために、そのバージョンを選択するように許可します。

11．再ライセンス

「大勢の複数著者協働サイト」（または「MMC サイト」）は、著作権の作品を公開して任意のワールド・ワイド・ウェブ・サーバーを意味し、また、それらの作品を編集するために誰のための著名な施設を提供します。誰もが編集できることを公共の wiki は、サーバーの一例です。サイトに含まれる「大勢の複数著者協働」（または「MMC」という。）を MMC サイトで公開著作権保護作品の任意のセットを意味します。

「CC-BY-SA」は、クリエイティブ・コモンズ・コーポレーション（カリフォルニア州サンフランシスコに主たる営業所を持つ非営利企業）よって発行されたクリエイティブ・コモンズ表示 - 継承 3.0 ライセンス、ならびにその同じ組織によって公開されたそのライセンスの将来のコピーレフトのバージョンを意味します。

「組み込む」とは、公開または文書を、全体的または部分的に、別の文書の一部として再発行することを意味します。

MMC は、それが本ライセンスの下でライセンスされている場合、「再ライセンスの対象」であり、すべての作品は、MMC に最初にこの MMC 以外の場所に本ライセンスの下で公開され、その後、全体的にまたは部分的に組み込まれたものならば、(1) 無ましましたテキスト又は不変セクションをカバーし、そして (2) このようにして前 2008 年 11 月 1 日に組み込まれました。

MMC サイトの運営者は、2009 年 8 月 1 日前の任意の時点で、同じサイト上の CC-BY-SA の下のサイトに含まれる MMC を再発行 MMC が再ライセンスの対象となり提供することがあります。

補遺：あなたの文書のために、このライセンスを使用する方法

文書内のライセンスのコピーを含め、書かれている文書で、このライセンスを使用すると、ちょうど題扉の後に、次の著作権およびライセンス通知を配置するには：

```
Copyright © YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

あなたは不変のセクションを持っている場合は、フロントカバーテキストとバックカバーテキストは、「テキスト... と。」置き換えます これに伴い：

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

あなたがカバーテキストのない不変のセクション、または 3 の他のいくつかの組み合わせを持っている場合は、状況に合わせて、これらの二つの選択肢を混ぜ合わせます。

文書中にプログラムコードの自明でない例が含まれている場合は、フリーソフトウェアでの使用を可能とするために、GNU 一般公衆利用許諾契約書として、フリーソフトウェアライセンスの選択の下で並行して

これらの例をリリースすることをお勧めします。

31.3 付録 C: QGIS のファイル形式

31.3.1 QGS/QGZ - QGIS プロジェクトファイル形式

QGS 形式は、QGIS プロジェクトを保存するための XML 形式です。 **QGZ** 形式は、QGS ファイルと QGD ファイルを含む圧縮（zip）アーカイブです。 **QGD** ファイルは、プロジェクトの補助データを含む qgis プロジェクトの関連する sqlite データベースです。補助データがない場合、QGD ファイルは空になります。

QGIS ファイルには、QGIS プロジェクトの保存に必要なすべてのものが含まれています：

- プロジェクトのタイトル
- プロジェクトの CRS
- レイヤツリー
- スナップ設定
- 関係
- マップキャンバスの範囲
- プロジェクトモデル
- 凡例
- mapview ドック（2D および 3D）
- 基になるデータセット（データソース）へのリンクを持つレイヤ、および範囲、SRS、結合、スタイル、レンダラ、ブレンドモード、不透明度などのその他のレイヤプロパティ。
- プロジェクトのプロパティ

以下の図は、QGS ファイルの最上位タグと展開された `ProjectLayers` タグを示しています。

```
−<qgis version="3.4.13-Madeira" projectname="">
    <homePath path=""/>
    <title/>
    <autotransaction active="0"/>
    <evaluateDefaultValues active="0"/>
    <trust active="0"/>
  +<projectCrs></projectCrs>
  +<layer-tree-group></layer-tree-group>
  +<snapping-settings tolerance="12" unit="1" enabled="0" type="1" mode="2" intersection-snapping="0">
    </snapping-settings>
    <relations/>
  −<mapcanvas name="theMapCanvas" annotationsVisible="1">
      <units>meters</units>
    +<extent></extent>
      <rotation>0</rotation>
    +<destinationsrs></destinationsrs>
      <rendermaptile>0</rendermaptile>
    </mapcanvas>
    <projectModels/>
  +<legend updateDrawingOrder="true"></legend>
    <mapViewDocks/>
    <mapViewDocks3D/>
  +<projectlayers></projectlayers>
  +<layerorder></layerorder>
  +<properties></properties>
    <visibility-presets/>
    <transformContext/>
  +<projectMetadata></projectMetadata>
    <Annotations/>
    <Layouts/>
  </qgis>
```

図 31.1: QGS ファイル中のトップレベルのタグ

```
−<projectlayers>
 −<maplayer styleCategories="AllStyleCategories" readOnly="0" autoRefreshTime="0" autoRefreshEnabled="0" refreshOnNotifyEnabled="0" maxScale="0"
  geometry="Polygon" labelsEnabled="0" type="vector" simplifyDrawingHints="1" hasScaleBasedVisibilityFlag="0" simplifyDrawingTol="1"
  simplifyMaxScale="1" minScale="1e+8" simplifyAlgorithm="0" simplifyLocal="1" refreshOnNotifyMessage="">
  +<extent></extent>
   <id>watersheds_b62efa19_8809_4406_b6ec_2951ac4c94c5</id>
  −<datasource>
     ./QGIS-Training-Data-2.0/exercise_data/processing/generalize/watersheds.shp
   </datasource>
  +<keywordList></keywordList>
   <layername>watersheds</layername>
  +<srs></srs>
  +<resourceMetadata></resourceMetadata>
   <provider encoding="UTF-8">ogr</provider>
   <vectorjoins/>
   <layerDependencies/>
   <dataDependencies/>
   <legend type="default-vector"/>
   <expressionfields/>
  +<map-layer-style-manager current="default"></map-layer-style-manager>
   <auxiliaryLayer/>
  +<flags></flags>
  +<renderer-v2 symbollevels="0" enableorderby="0" type="singleSymbol" forceraster="0"></renderer-v2>
  +<customproperties></customproperties>
   <blendMode>0</blendMode>
   <featureBlendMode>0</featureBlendMode>
   <layerOpacity>1</layerOpacity>
  +<SingleCategoryDiagramRenderer diagramType="Histogram" attributeLegend="1"></SingleCategoryDiagramRenderer>
  +<DiagramLayerSettings priority="0" linePlacementFlags="18" dist="0" showAll="1" placement="1" obstacle="0" zIndex="0"></DiagramLayerSettings>
  +<geometryOptions removeDuplicateNodes="0" geometryPrecision="0"></geometryOptions>
  +<fieldConfiguration></fieldConfiguration>
  +<aliases></aliases>
   <excludeAttributesWMS/>
   <excludeAttributesWFS/>
  +<defaults></defaults>
  +<constraints></constraints>
  +<constraintExpressions></constraintExpressions>
   <expressionfields/>
  +<attributeactions></attributeactions>
  +<attributetableconfig actionWidgetStyle="dropDown" sortExpression="" sortOrder="0"></attributetableconfig>
  +<conditionalstyles></conditionalstyles>
   <editform tolerant="1"/>
   <editforminit/>
   <editforminitcodesource>0</editforminitcodesource>
   <editforminitfilepath/>
  +<editforminitcode></editforminitcode>
   <featformsuppress>0</featformsuppress>
   <editorlayout>generatedlayout</editorlayout>
  +<editable></editable>
  +<labelOnTop></labelOnTop>
   <widgets/>
   <previewExpression>ID</previewExpression>
   <mapTip/>
 </maplayer>
</projectlayers>
```


図 31.2: QGS ファイルの展開された最上位の ProjectLayers タグ

31.3.2 QLR - QGIS レイヤー定義ファイル

レイヤ定義ファイル（QLR）は、レイヤの QGIS スタイル情報に加えて、レイヤデータソースへのポインタを含む XML ファイルです。

このファイルの使用例は簡単です。データソースを開き、関連するすべてのスタイル情報を取り込むための単一のファイルを用意することです。 QLR ファイルを使用すると、簡単に開けるファイルで基になるデータソースをマスクすることもできます。

QLR の使用例は、MS SQL レイヤを開くためのものです。 MS SQL 接続ダイアログに移動して、接続、選択、ロード、最後にスタイルを設定する代わりに、必要なすべてのスタイルが含まれる正しい MS SQL レイヤを指す.qlr ファイルを追加するだけです。

将来、.qlr ファイルは複数のレイヤへの参照を保持する可能性があります。

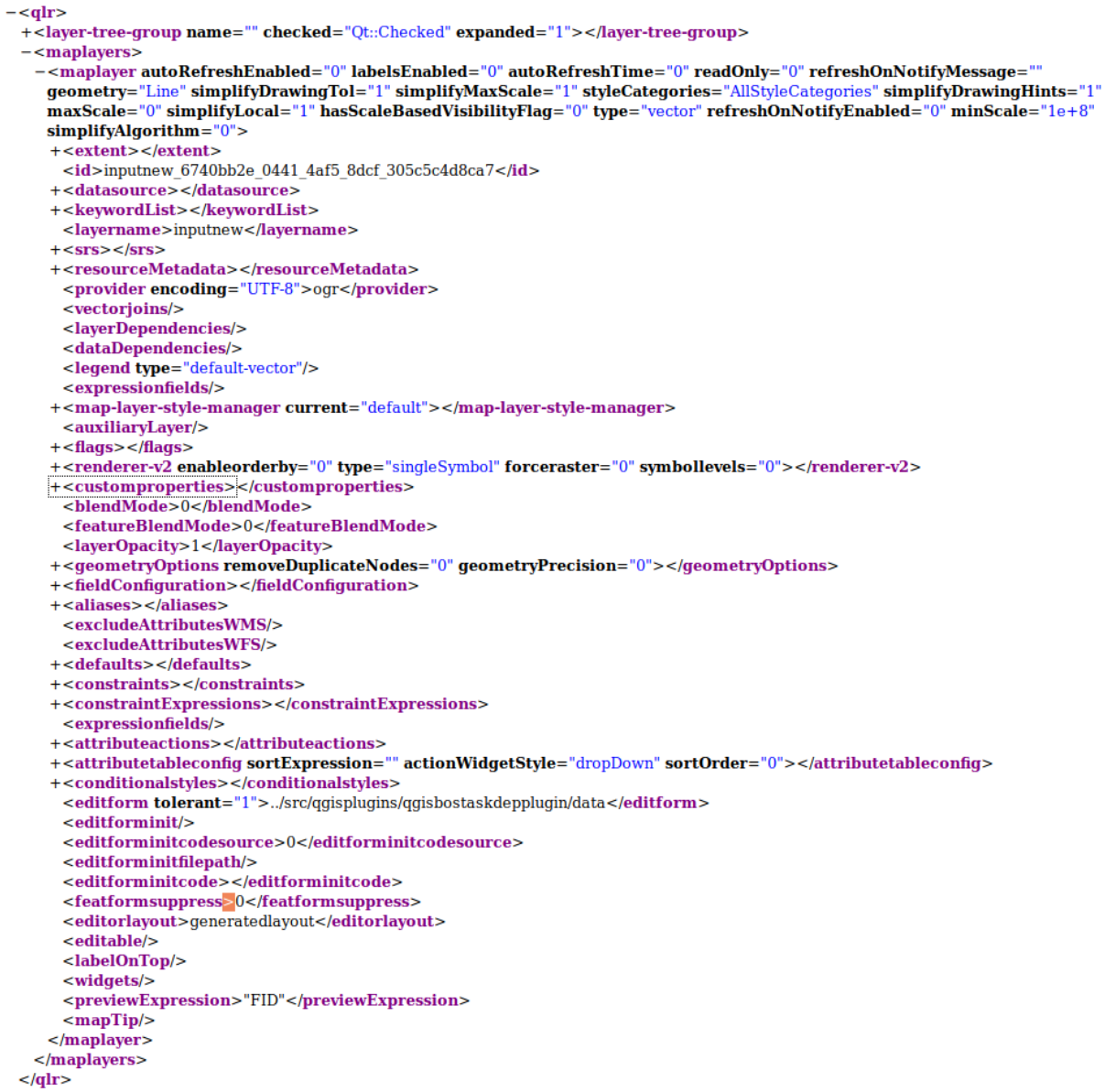


図 31.3: QLR ファイルのトップレベルのタグ

31.3.3 QML-QGIS スタイルファイル形式

QML は、レイヤのスタイルを保存するための XML 形式です。

QML ファイルには、シンボル定義、サイズと回転、ラベリング、不透明度、ブレンドモードなどを含む地物ジオメトリのレンダリングのために QGIS が処理できるすべての情報が含まれています。

以下の図は QML ファイルの最上位タグを示しています（ `renderer_v2` とその `symbol` タグのみが展開されています）。

```
−<qgis version="3.4.13-Madeira" styleCategories="AllStyleCategories" readOnly="0" maxScale="0"
 labelsEnabled="0" simplifyDrawingHints="1" hasScaleBasedVisibilityFlag="0" simplifyDrawingTol="1"
 simplifyMaxScale="1" minScale="1e+8" simplifyAlgorithm="0" simplifyLocal="1">
 +<flags></flags>
 −<renderer-v2 symbollevels="0" enableorderby="0" type="singleSymbol" forceraster="0">
   −<symbols>
     +<symbol clip_to_extent="1" name="0" alpha="1" type="fill" force_rhr="0"></symbol>
    </symbols>
    <rotation/>
    <sizescale/>
  </renderer-v2>
 +<customproperties></customproperties>
  <blendMode>0</blendMode>
  <featureBlendMode>0</featureBlendMode>
  <layerOpacity>1</layerOpacity>
 +<SingleCategoryDiagramRenderer diagramType="Histogram" attributeLegend="1">
  </SingleCategoryDiagramRenderer>
 +<DiagramLayerSettings priority="0" linePlacementFlags="18" dist="0" showAll="1" placement="1"
  obstacle="0" zIndex="0">
  </DiagramLayerSettings>
 +<geometryOptions removeDuplicateNodes="0" geometryPrecision="0"></geometryOptions>
 +<fieldConfiguration></fieldConfiguration>
 +<aliases></aliases>
  <excludeAttributesWMS/>
  <excludeAttributesWFS/>
 +<defaults></defaults>
 +<constraints></constraints>
 +<constraintExpressions></constraintExpressions>
  <expressionfields/>
 +<attributeactions></attributeactions>
 +<attributetableconfig actionWidgetStyle="dropDown" sortExpression="" sortOrder="0">
  </attributetableconfig>
 +<conditionalstyles></conditionalstyles>
  <editform tolerant="1"/>
  <editforminit/>
  <editforminitcodesource>0</editforminitcodesource>
  <editforminitfilepath/>
 +<editforminitcode></editforminitcode>
  <featformsuppress>0</featformsuppress>
  <editorlayout>generatedlayout</editorlayout>
 +<editable></editable>
 +<labelOnTop></labelOnTop>
  <widgets/>
  <previewExpression>ID</previewExpression>
  <mapTip/>
  <layerGeometryType>2</layerGeometryType>
</qgis>
```

図 31.4: QML ファイルのトップレベルタグ（symbol タグを持つ renderer_v2 タグのみが展開されています）

31.4 付録 D：QGIS R スクリプト構文

Matteo Ghetta の寄稿 - Scuola Superiore Sant'Anna による資金提供

プロセシングで R スクリプトを書くのは、特殊な構文のため、少しコツがいります。

プロセシングの R スクリプトは、まず **入力** と **出力** を定義し、それぞれの前に二重のハッシュ文字 (##) を付けます。

入力の前に、アルゴリズムを配置するグループを指定できます。グループがすでに存在する場合はそこにアルゴリズムが追加されます。存在しない場合は、グループが作成されます。以下の例では、グループの

名前は *My group* です：

```
##My Group=group
```

31.4.1 入力

すべての入力データとパラメータは指定する必要があります。入力は何種類かあります：

- ベクタ: `##Layer = vector`
- ベクタのフィールド: `##F = Field Layer`（ここで *Layer* はフィールドが属する入力ベクタレイヤの名前です）
- ラスタ: `##r = raster`
- 表: `##t = table`
- 数値: `##Num = number`
- 文字列: `##Str = string`
- ブール値: `##Bol = boolean`
- ドロップダウンメニューの要素。項目はセミコロン ; で区切らなければなりません: `##type=selection point;lines;point+lines`

31.4.2 出力

入力と同じように、各出力は、スクリプトの先頭で定義する必要があります。

- ベクタ: `##output= output vector`
- ラスタ: `##output= output raster`
- 表: `##output= output table`
- プロット: `##output_plots_to_html`（以前の版では##showplots）
- R の出力を **結果ビューアー** に表示するには、出力を表示したいコマンドの前に > を置きます。

31.4.3 QGIS R スクリプトの構文の概要

たくさんの入出力パラメータ型が用意されています。

入力パラメータの型

パラメータ	構文の例	返すオブジェクト
vector	Layer = vector	sf オブジェクト（または、##load_vector_using_rgdal が指定されたときは SpatialDataFrame オブジェクト）
vector point	Layer = vector point	sf オブジェクト（または、##load_vector_using_rgdal が指定されたときは SpatialDataFrame オブジェクト）
vector line	Layer = vector line	sf オブジェクト（または、##load_vector_using_rgdal が指定されたときは SpatialDataFrame オブジェクト）
vector polygon	Layer = vector polygon	sf オブジェクト（または、##load_vector_using_rgdal が指定されたときは SpatialPolygonsDataFrame オブジェクト）
multiple vector	Layer = multiple vector	sf オブジェクト（または、##load_vector_using_rgdal が指定されたときは SpatialDataFrame オブジェクト）
table	Layer = table	csv からのデータフレーム変換、 `read.csv` 関数のデフォルトオブジェクト
field	Field = Field Layer	選択されたフィールドの名前、例えば `"Area"`
raster	Layer = raster	RasterBrick オブジェクト、 `raster` パッケージのデフォルトオブジェクト
multiple raster	Layer = multiple raster	RasterBrick オブジェクト、 `raster` パッケージのデフォルトオブジェクト
number	N = number	選択された整数または浮動小数点数
string	S = string	ボックスに追加された文字列
longstring	LS = longstring	文字列がボックスに追加され、通常の文字列より長くなる可能性があります
selection	S = selection first;second;third	ドロップダウンメニューで選択した選択項目の文字列
crs	C = crs	結果として選ばれた CRS の文字列。形式は `"EPSG:4326"`
extent	E = extent	`raster` パッケージの範囲オブジェクトでは、値を `E@xmin` として抽出できます
point	P = point	地図をクリックすると点の座標が得られます
file	F = file	選択されたファイルのパス、例えば「/home/matteo/file.txt」
folder	F = folder	選択されたフォルダのパス、例えば「/home/matteo/Downloads」

パラメータは **OPTIONAL** つまり、無視できるようにできます。

入力をオプションとして設定するためには、その入力の **前** に文字列 `optional` を追加します、例えば:

```
##Layer = vector
##Field1 = Field Layer
##Field2 = optional Field Layer
```

出力パラメータの種類

パラメータ	構文の例
vector	Output = output vector
raster	Output = output raster
table	Output = output table
file	Output = output file

注釈: **プロセシング結果ビューア** からプロットを png として保存することも、アルゴリズムインターフェースから直接プロットを保存することもできます。

スクリプト本体

スクリプト本体は R 構文に従い、 **Log** パネルはスクリプトに問題がある場合に役立ちます。

スクリプトですべての追加ライブラリを読み込む必要があることを **覚えてください**:

```
library(sp)
```

31.4.4 例

ベクタ出力の例

入力レイヤの範囲からランダムな点を作成するオンラインコレクションからのアルゴリズムを見てみましょう:

```
##Point pattern analysis=group
##Layer=vector polygon
##Size=number 10
##Output=output vector
library(sp)
spatpoly = as(Layer, "Spatial")
pts=spsample(spatpoly,Size,type="random")
spdf=SpatialPointsDataFrame(pts, as.data.frame(pts))
Output=st_as_sf(spdf)
```

説明（スクリプトの行ごと）:

1. `Point pattern analysis`（ポイントパターン分析） はアルゴリズムのグループ
2. `Layer` は入力 **ベクタ** レイヤ
3. `Size` は、**numerical（数値）** パラメタであり、デフォルト値は 10

4. Output はアルゴリズムによって作成される **ベクタ** レイヤ
5. library(sp) は、**sp** ライブラリを読み込む
6. spatpoly = as(Layer, "Spatial") は、sp オブジェクトに翻訳する
7. sp ライブラリの spsample 関数を呼び出し、上で定義した入力（Layer と Size）を使って実行する
8. SpatialPointsDataFrame 関数を使って *SpatialPointsDataFrame* オブジェクトを生成する
9. st_as_sf 関数を使って出力ベクタレイヤを生成する

それでおしまい！あとは QGIS 凡例中にあるベクタレイヤでアルゴリズムを実行し、ランダムなポイントの数を選択するだけです。結果のレイヤが地図に追加されます。

ラスター出力の例

次のスクリプトでは、「automap」R パッケージの「autoKrige」関数を使用して入力ポイントベクタレイヤの指定されたフィールドから補間値のラスタマップを作成するために、基本的な通常のクリギングを実行します。最初にクリギングモデルを計算し、次にラスタを作成します。ラスターはラスタ R パッケージの raster 関数で作成されます:

```
##Basic statistics=group
##Layer=vector point
##Field=Field Layer
##Output=output raster
##load_vector_using_rgdal
require("automap")
require("sp")
require("raster")
table=as.data.frame(Layer)
coordinates(table)= ~coords.x1+coords.x2
c = Layer[[Field]]
kriging_result = autoKrige(c~1, table)
prediction = raster(kriging_result$krige_output)
Output<-prediction
```

##load_vector_using_rgdal を使うと、入力ベクタレイヤは SpatialDataFrame オブジェクトとして使用できるようになり、sf オブジェクトから変換する必要がなくなります。

テーブル出力の例

出力がテーブルファイル（CSV）になるように 要約統計 アルゴリズムを編集してみましょう。

スクリプトの本文は以下の通りです:

```
##Basic statistics=group
##Layer=vector
##Field=Field Layer
##Stat=Output table
Summary_statistics<-data.frame(rbind(
    sum(Layer[[Field]]),
    length(Layer[[Field]]),
    length(unique(Layer[[Field]])),
    min(Layer[[Field]]),
    max(Layer[[Field]]),
    max(Layer[[Field]])-min(Layer[[Field]]),
    mean(Layer[[Field]]),
    median(Layer[[Field]]),
    sd(Layer[[Field]])),
  row.names=c("Sum:","Count:","Unique values:","Minimum value:","Maximum value:",
↪"Range:","Mean value:","Median value:","Standard deviation:"))
colnames(Summary_statistics)<-c(Field)
Stat<-Summary_statistics
```

３行目は入力に **ベクターフィールド** を指定し、４行目は出力テーブルであるべきであるアルゴリズムを伝えます。

最後の行は、スクリプトで作成された Stat オブジェクトを取得し、 csv テーブルに変換します。

コンソール出力の例

前の例を使用して、テーブルを作成する代わりに、 **結果ビューア** で結果を印刷できます:

```
##Basic statistics=group
##Layer=vector
##Field=Field Layer
Summary_statistics<-data.frame(rbind(
sum(Layer[[Field]]),
length(Layer[[Field]]),
length(unique(Layer[[Field]])),
min(Layer[[Field]]),
max(Layer[[Field]]),
max(Layer[[Field]])-min(Layer[[Field]]),
mean(Layer[[Field]]),
median(Layer[[Field]]),
```

(次のページに続く)

(前のページからの続き)

```
sd(Layer[[Field]])),row.names=c("Sum:","Count:","Unique values:","Minimum value:",
↪"Maximum value:","Range:","Mean value:","Median value:","Standard deviation:"))
colnames(Summary_statistics)<-c(Field)
>Summary_statistics
```

スクリプトは、2 つの編集を除いて、上のものとまったく同じです：

1. 出力は指定されていません（4 行目は取り除かれています）
2. 最後の行は > で始まるので、プロセシングに結果ビューアからオブジェクトを利用できるようにするよう指示しています

プロットと例

プロットを作成するには、次のスクリプトのように ##output_plots_to_html パラメータを使用する必要があります:

```
##Basic statistics=group
##Layer=vector
##Field=Field Layer
##output_plots_to_html
####output_plots_to_html
qqnorm(Layer[[Field]])
qqline(Layer[[Field]])
```

このスクリプトは、ベクタレイヤ（Layer）のフィールド（Field）を入力として使用し、*QQ プロット*（分布の正規性をテストする）を作成します。

プロットは自動的にプロセシング **結果ビューア** に追加されます。

31.5 付録 E：QGIS アプリケーションのネットワーク接続

以下のリストは、QGIS が実行する事前定義または自動的なネットワーク接続の一覧です。接続前にユーザーからのアクションを必要とするためユーザーが接続開始するものもあれば、自動的に接続が行われるものもあります。

名前	目的	モード	サーバ	サーバに送られる情報	サーバに保存される情報
qgis.org					
Python API のヘルプ	PyQGIS ドキュメントの閲覧	ユーザー起動	https://qgis.org/pyqgis/%1/search.html?q=%2	IP、QGIS のバージョン、OS 情報	サーバログの IP
version.qgis					
新しいバージョンのチェック	新しいバージョンの QGIS が利用可能となった場合にユーザーに通知する	自動	https://version.qgis.org	IP、QGIS のバージョン、OS 情報	サーバログの IP
feed.qgis					
QGIS のフィード	フィードから QGIS のニュースを取得する	自動	https://feed.qgis.org	IP、QGIS のバージョン、言語コード、最終ダウンロードのタイムスタンプ、OS 情報	サーバーログ中の IP; QGIS のバージョン、OS、IP が集計され、統計情報の収集に使用される。
plugins.qgis.					
プラグイン更新の確認	プラグインのアップデートを通知する	ユーザー起動 / 自動（設定を変更可能）	https://plugins.qgis.org	IP、QGIS のバージョン、OS 情報	サーバログの IP
プラグインのリスト	プラグインのリストを取得する	ユーザー起動	https://plugins.qgis.org	IP、QGIS のバージョン、OS 情報	サーバログの IP
プラグインのインストール	プラグインのパッケージをダウンロードしインストールする	ユーザー起動	https://plugins.qgis.org	IP、QGIS のバージョン、OS 情報	プラグインのダウンロード回数が 1 増加する
スタイル	ユーザーが投稿したスタイルのリスト	ユーザー起動	https://plugins.qgis.org/styles	IP、QGIS のバージョン、OS 情報	ダウンロード回数が 1 増加する
サードパーティ					
地形データ	3D ビューのための DEM を作成する	ユーザー起動	https://s3.amazonaws.com/elevation-tiles-pro terrarium/{z}/{x}/	IP、QGIS のバージョン、OS 情報	Amazon のサービス利用規約参照
Google Map Geocode	ジオコーディングサービス	ユーザー起動	https://maps.googleapis.com/maps/api/geocode/json	IP、QGIS のバージョン、OS 情報	google map API のサービス利用規約参照
Nomi-	ジオコーディン	ユーザー	https:	IP、QGIS のバージ	

第32章　文献とWeb参照

GDAL-SOFTWARE-SUITE. Geospatial data abstraction library. https://gdal.org, 2013.

GRASS-PROJECT. 地理的資源分析支援システム (Geographic resource analysis support system) https://grass.osgeo.org, 2013.

NETELER, M., AND MITASOVA, H. オープンソース GIS グラス アプローチ (Open source gis: A grass gis approach, 2008)

OGR-SOFTWARE-SUITE. Geospatial data abstraction library. https://gdal.org, 2013.

OPEN-GEOSPATIAL-CONSORTIUM. Web map service (1.1.1) implementation specification. https://portal.ogc.org/files/?artifact_id=1081&version=1&format=pdf, 2002.

OPEN-GEOSPATIAL-CONSORTIUM. Web map service (1.3.0) implementation specification. https://portal.ogc.org/files/?artifact_id=14416&format=pdf, 2004.

POSTGIS-PROJECT. PostgreSQL の空間サポート。 http://www.refractions.net/products/postgis/, 2013.